THESIS

IMPLEMENTING A PATTERNLESS INTRUSION DETECTION SYSTEM; A METHODOLOGY FOR ZIPPO

by

Vonda L. Olsavsky

September 2005

Primary Advisor:	John McEachen
Associate Advisor:	Alex Bordetsky

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2005		**3. REPORT TYPE AND DATES COVERED** Master's Thesis
4. TITLE AND SUBTITLE: Title (Mix case letters) Implementing a Patternless Intrusion Detection System; A Methodology for Zippo			**5. FUNDING NUMBERS**
6. AUTHOR(S) Vonda L. Olsavsky			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

A methodology for the implementation of Zippo, a patternless intrusion detection system is presented in this thesis. This methodology approaches the implementation in a holistic manner to include the administrative and operational tasks necessary for ensuring proper preparation for Zippo's use. Prior to implementing and using Zippo, a basic understanding of TCP/IP and intrusion detection systems is needed and these topics are presented in broad detail. The origin of Zippo starts with the creation of Therminator, which is discussed in detail. The architecture and configuration of Zippo are based on those of Therminator and understanding the ideas of buckets and balls, thermal canyons and towers, decision trees, slidelength and windowlength and initial and boundary conditions are paramount to understanding the Zippo application. To successfully implement Zippo, other network factors must be attended to including the topology, organizational policies and the security plan. Once these factors are addressed, Zippo can be optimally configured to successfully be installed on a network. Finally, previous research done on Zippo yielded decision trees and thermal canyons pertaining to protocol specific threats that are presented to familiarize the reader with Zippo's visual representation of malicious or anomalous behavior.

14. SUBJECT TERMS Network Security, Intrusion Detection, Zippo, Therminator, Patternless Intrusion Detection Systems			**15. NUMBER OF PAGES** 125
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

IMPLEMENTING A PATTERNLESS INTRUSION DETECTION SYSTEM; A METHODOLOGY FOR ZIPPO

Vonda L. Olsavsky
Lieutenant, United States Navy
B.S., Clemson University, 1998

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 2005

Author:	Vonda L. Olsavsky
Approved by:	John McEachen Primary Thesis Advisor
	Alex Bordetsky Associate Thesis Advisor
	Dan Boger Chairman, Department of Information Systems and Technology

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

A methodology for the implementation of Zippo, a patternless intrusion detection system is presented in this thesis. This methodology approaches the implementation in a holistic manner to include the administrative and operational tasks necessary for ensuring proper preparation for Zippo’s use. Prior to implementing and using Zippo, a basic understanding of TCP/IP and intrusion detection systems is needed and these topics are presented in broad detail. The origin of Zippo starts with the creation of Therminator, which is discussed in detail. The architecture and configuration of Zippo are based on those of Therminator and understanding the ideas of buckets and balls, thermal canyons and towers, decision trees, slidelength and windowlength and initial and boundary conditions are paramount to understanding the Zippo application. To successfully implement Zippo, other network factors must be attended to including the topology, organizational policies and the security plan. Once these factors are addressed, Zippo can be optimally configured to successfully be installed on a network. Finally, previous research done on Zippo yielded decision trees and thermal canyons pertaining to protocol specific threats that are presented to familiarize the reader with Zippo’s visual representation of malicious or anomalous behavior.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First of all, I would like to thank my husband Tom for his unconditional love and support and for his belief in me and my abilities. I would also like to thank Dr. John McEachen for giving me the opportunity to work with his group, for his guidance and for what seemed to be his endless supply of patience. And last but definitely not least, I would like to thank Dr. Alex Bordetsky for his amazing intelligence, enthusiasm for networks and the appreciation he shows to those around him.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. COMPUTER NETWORK DEFENSE

In today’s network-centric society, with the prevalence of e-business and electronic communications, the quest for secure networks has been brought to the front lines of the technology battlefield. Networks are found in every facet of daily life. The world’s financial investment market relies on networks. The backbone of global communications and transportation is a network. Networks form the infrastructure of the world’s largest organizations to the smallest “mom and pop” businesses.

With the proliferation of networks comes the proliferation of exploitation. As the number of networks rise, so does the number of malicious hackers trying to “illegally” gain access to the systems to carryout such behavior as reconnaissance, exploitation, reinforcement to elevate privileges, consolidation to communicate undetectably with the system and pillaging [BEJT-05]. This means that system administrators have an ever increasing need to ensure security of their networks and safety of their data. In much the same way a military must develop a strategy to defend its assets and its perimeter, so too must system administrators develop a strategy to defend the boundaries of their networks. Network defense in depth is a term that has been adopted to address this issue.

A good defense in depth contains numerous security programs and procedures to address malicious behavior and misuse of a network system. Physical security as well as network security should be addressed for the proper defense of a system. Applying only one defensive measure to a computer network would be analogous to the little Dutch boy being able to plug only one hole in the dike, allowing water to breach all other entrances, thereby destroying it. Defense in depth requires a thorough security plan that addresses all facets of computer network security to include the perimeter, the internal system and the humans in the loop.

There are hundreds of products available in the commercial market to aid in thwarting network attacks. Networks can use proxy servers to “hide” real servers that service an organization. Firewalls can be configured to allow traffic from defined sets of IP addresses and disallow traffic from others. Another tool available to support a strong defense in depth posture is a patternless intrusion detection system.

B. PATTERNLESS INTRUSION DETECTION SYSTEMS (PIDS)

Intrusion detection systems (IDS) provide a fairly robust computer defense mechanism that searches incoming traffic for defined signatures or aberrant behavior. There are numerous types of IDS on the market today. Signature-based, anomaly, host- or network-based systems make up the majority of commercial IDS available. Patternless intrusion detection systems, or PIDS, are the newest technologies on the block. PIDS provide visibility into the traffic that exists on a network, giving system administrators another tool to monitor not only the health of their network but also the effectiveness of the other network defense systems they are using.

In a PIDS, the program relies on neither signatures of known malicious activity nor human behavior models to seek out anomalous behavior: instead, it bases its interpretation of network behavior on a statistical analysis of network traffic and knowledge of network policy. The traffic is modeled and available for reviewing or viewing in real-time by the system administrator. An example of a PIDS is the Therminator program created by Stephen Donald and Robert McMillen in conjunction with Dr. David Ford, the Naval Postgraduate School and the National Security Agency [DONA-01].

1. Therminator

The Therminator is a network-based PIDS that relies on the principles of thermodynamic statistics to model network traffic. It uses categories and conversations as the underlying base of the network model. A real-time 3-dimensional model of a network’s traffic is created by the program and displayed in a graphical user interface (GUI) that is easy for the administrator to view and interpret. A further discussion on the evolution of Therminator will be presented in Chapter IV.

2. Zippo

Subsequent to the development of Therminator was the creation of Zippo. The Therminator program was not created as a final system to be used for long-term installation or for demonstration tasks [ZACH-04]. Zippo was created to address these short-comings of Therminator.

Dr. John Zachary and his team at the University of South Carolina were contracted to develop Zippo as a more robust software implementation of the Therminator model. Zippo primarily addresses the need for longer execution times, higher traffic volume and faster processing and the ability to be used across numerous types of platforms and network configurations [ZACH-04]. A further description of Zippo will be presented in Chapter V.

3. Network Thermal Vistas

Network Thermal Vistas (NTV) is an innovative and robust patternless intrusion detection system developed by Secure Cognition, Inc. Network Thermal Vistas uses the patented Therminator technology, licensed from the National Security Agency, to provide a commercially available version of the Therminator application that has been improved to be more scalable and, like Therminator and Zippo, provides an intuitive 3-dimensional display that shows real-time network state changes. With NTV, Secure Cognition Inc. has successfully marketed the Therminator technology, which has been readily adopted by the entertainment industry, and continues to add to the system's successes and prove that the Patternless Intrusion Detection concept adds an original layer to a network's defense in depth [SECCOG-04].

C. THESIS MOTIVATION

Network administrators are given the weighty task of defending their organizations' networks against malicious activity and even activity caused unknowingly by trusted users (e.g. Trojan horses installed accidentally by installation of a useful program). This is a very substantial task considering a system administrator is responsible for finding all the holes in his network and a malicious hacker need only find one. One of the tools that can greatly enhance a system administrator’s ability to safeguard their network is the Zippo patternless intrusion detection system.

The purpose of this thesis is to develop a methodology for implementing Zippo on any given network. The methodology will encompass the steps necessary to properly use Zippo as well as assets that need to already be in place prior to implementing Zippo. Additionally, the first section of this thesis cursorily examines the protocols that are used in network communications so the reader understands how an IDS works prior to employing one.

During the research phase of this thesis, the author worked with an organization already using Therminator. The network experts assigned to the organization were interviewed in great depth and the implementation methodology created by the author is heavily influenced by the work done with these individuals and their network. Because of confidentiality and security concerns, the actual organization, its members and their network of interest cannot be identified.

The ultimate motivation behind this thesis is to create a methodology for an organization to take the next step in patternless intrusion detection and implement Zippo on their network systems. It is the author’s intent that after reading this thesis the reader will have the necessary knowledge to be able to adequately understand what an IDS is, how it works, what assets need to be in place prior to the implementation of the IDS, how to properly configure Zippo and how to properly monitor/translate the visual display Zippo presents. This thesis is not the panacea to computer security in the same manner that one simple IDS is not the panacea to defense in depth; it is merely a tool that provides the reader with one more level of security to properly safeguard important data against malfeasants.

D. THESIS ORGANIZATION

The next four chapters of this thesis develop the foundation for understanding what an IDS is and how it works. Chapter II is a brief introduction (or review for some) of Transmission Control Protocol and Internet Protocol (TCP/IP). Chapter III will discuss what an IDS is and the different types of IDS available. Chapter IV is an in-depth look at the Therminator IDS, as the author feels it is very important to understand how a system works prior to using it. Chapter V concludes the introductory section with a description of Zippo, its evolution and its capabilities.

Chapters VI – VIII present the assets that should be in place prior to implementing Zippo. These include a network topology with defined services and system inventory, the organizational policies and procedures with regard to such topics as accessibility, usage, training, etc., and lastly an organization’s security plan. These chapters will present what the author thinks is the preferred architecture for policies and plans and provide guidance on developing them and on developing a network topology.

Chapters IX and X deal specifically with the Zippo PIDS. These chapters will outline for the reader an approach to configuring Zippo for a specific network based on previous research conducted at the Naval Postgraduate

School and will look at the process of monitoring Zippo during implementation.

Lastly, Chapter XI will provide a summary of the steps introduced throughout the thesis and will discuss useful areas of future research.

E. SUMMARY

With the increase in size and reach of the Internet, and as more and more businesses and organizations have come to rely on networking, so too has the number of malicious users increased. It is much easier for a hacker to gain access to a network through one small hole, than it is for a system administrator to protect a network with lots of holes. Instituting a defense in depth, using numerous methodologies, tools, policies and physical security layers aids a system administrator in protecting their network.

An Intrusion Detection System (IDS) is one layer in a defense in depth to protect a network against malicious behavior. In the recent past, a Patternless Intrusion Detection System was developed at the Naval Postgraduate School to be used as a part of a defense in depth. Therminator, as this tool is called, is just the beginning of numerous generations of PIDS to include the topic of this thesis, Zippo.

In this thesis, basic IDS will be discussed along with the foundation for understanding how Zippo works (i.e. TCP/IP). Additionally, this thesis will set out a methodology that any system administrator can follow for implementing Zippo on their own network. Lastly, this thesis will discuss the optimal way of configuring Zippo and how to interpret the feedback received from Zippo.

As stated before, Zippo is not a tool that can effectively protect an entire network on its own. It must be used in conjunction with other elements of computer network defense to include physical security, firewalls, auditing and above all else the human factor.

II. TRANSMISSION CONTROL PROTOCOL AND INTERNET PROTOCOL

A. CHAPTER OVERVIEW

This chapter provides a brief review of Transmission Control Protocol/Internet Protocol (TCP/IP) and how it pertains to IDS. It looks at the TCP/IP four-layer model and packet architecture and investigates IP addressing and TCP service ports which are two tools used in the configuration of the buckets in Zippo. Additionally, the following chapter touches on a number of other protocols and typical services that are of interest to system administrators trying to protect a network.

B. TCP/IP CONCEPTS

TCP and IP are protocols used for communication over one type of computer network. A protocol, in the simplest definition, is a set of rules that standardize how something is to be done. In the case of TCP/IP, standards are created so that systems can communicate with each other regardless of what kind of hardware they may be composed of. IDS generally rely on the TCP/IP suite because they provide the packet architecture and addressing format used for packets traversing the network.

One of the best ways to model the TCP/IP protocol and Internet communication is to use the four-layer TCP/IP Internet layered model. This model, as presented by [BEJT-04], is depicted in Figure 1 below and shows the different layers used to model the hierarchy of Internet communication.

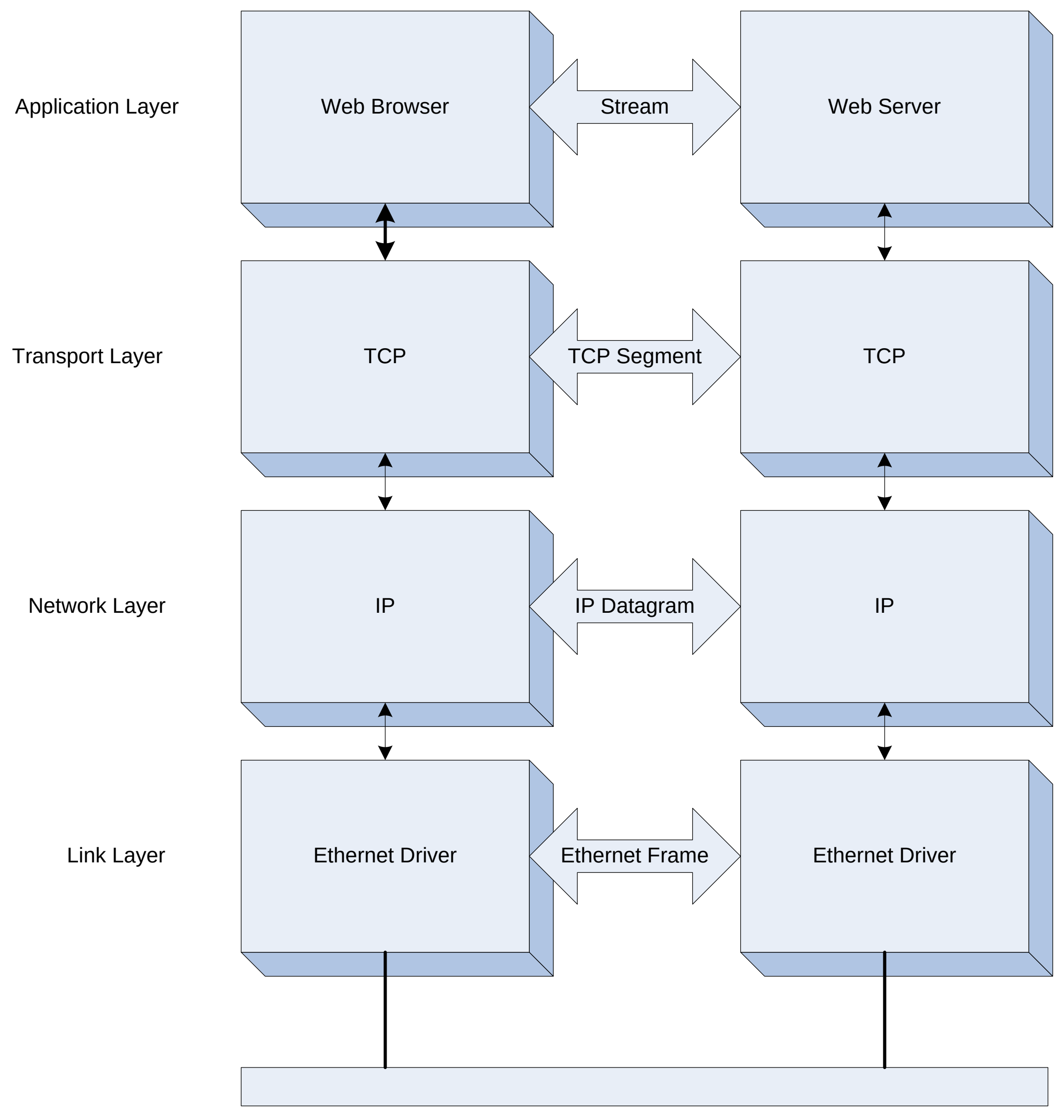

Figure 1. Four-Layer Internet Model

The application layer of the model is supported by the software installed on the end nodes of the communicating systems (the sender and receiver). The transport layer is governed by TCP (or User Datagram Protocol (UDP) in some instances) and is responsible for managing end-to-end communications between the sender and receiver. TCP is the reliable connection protocol while UDP does not guarantee delivery of packets. The network layer is responsible for getting packets from the source to the destination. Lastly, the link layer is the component that gets the data to the physical medium on which it will travel between the sender and receiver.   At each layer of the stack, the packet from the previous layer (also known as the payload) is wrapped up (or encapsulated) with the current layer’s header information.

The TCP/IP Internet model provides a good visual aid to assist in understanding how packets travel through the hierarchy and between the sender and receiver. Numerous other models exist, but the author has chosen this one as it gives a very simple view of TCP and IP which are the main protocols that most IDS are concerned with.

Theoretically, when the sender asks for information from the receiver and the receiver in turn responds, the request follows the following path: The sender originates the request in the application layer of their stack. The application layer packages the data from the application and any application protocols that are needed; this is referred to as the payload of the packet. The application layer tells the transport layer how the packet is to be sent (via either TCP or UDP) and the request works its way down the stack, getting wrapped up (or encapsulated) in the TCP and IP layers through the physical layer to the Ethernet and over to the receiver’s stack using IP addressing. The request works its way up the receiver’s stack to the application layer getting unwrapped in the TCP and IP layers as it goes. The application layer determines what course of action to take with regard to the request and if a response is necessary, the process starts over again and the receiver becomes the new sender.

The path above is simplified for easy understanding and because the following sections will discuss TCP/IP, encapsulation and addressing in greater depth.

C. TRANSMISSION CONTROL PROTOCOL

TCP is the main protocol used in the transport layer. TCP is a reliable connection-oriented protocol that provides feedback to the sender's stack ensuring proper delivery of the packets. UDP, which is also a protocol used in the transport layer, is connectionless and is an unreliable protocol that is used by applications that do not necessarily need to ensure the arrival of their packets to the recipients.

The TCP/UDP header applied at the transport layer includes two 16-bit port number fields that denote what port (between 0 and 65,535) the service desired should be found on (there are two as one is the source port and one is the destination port). There are many well-known ports that are universally accepted for specific services. Table 1 below presents some of the more well known ports.

Port Number	Service
7	Echo
20	ftp-data
21	ftp-control
22	SSH
23	Telnet
25	SMTP
53	DNS
79	Finger
80	HTTP
110	POP3
111	SunRPC
139	NetBIOS
443	SSL
514	RSH

Table 1. Common TCP Service Ports

The ports below 1024 are often referred to as the trusted ports and any port above 1024 is called an ephemeral port. Originally, the trusted ports were to be used by the system processes and the ephemeral ports could be used by any service. However, with the wide-spread explosion of the Internet's popularity, and the lack of security on it, the trusted ports are not necessarily to be trusted any longer.

Along with the services, it is important to understand the TCP flags in the header. The SYN flag, or synchronize flag, indicates the start of a TCP conversation. The ACK flag, or acknowledge flag, indicates acknowledgement or receipt of packets or a connection request. The FIN flag, or finish flag, is used to end a session between two hosts. The RST flag, or reset flag, re-synchronizes a connection between the systems. The PSH flag, or push flag, tells the sender to send data immediately vice having it "wait in line" to be sent.

The TCP connection created between two hosts is started by a three-way handshake. The initiating machine sends a SYN packet which signals to the receiver that it would like to start a conversation. The recipient replies with a SYN and an ACK (in a single packet) to acknowledge the original request. And lastly, the initiator replies with a final ACK. This is how all TCP conversations are started and once the three-way handshake is completed, a connection has been created between the two hosts.

The TCP transport model does every thing it can to ensure that packets are delivered to the recipient. Additionally, it makes certain that packets are placed into their proper order upon arrival so there is a sense of order to the message. TCP does this by requiring an acknowledgement, or ACK, whenever a packet is received. Therefore, in a TCP "conversation" there are many packets going back and forth between the participants, not just in one direction. UDP, on the other hand, simply packages the packets and sends them off to the recipient in hopes that they are received. It is much faster than TCP, but not nearly as reliable.

TCP is important to IDS as it is a very often used protocol on the Internet. Additionally, the service ports are important as often malicious behavior can be spotted by system administrators when the wrong types of packets are sent to the wrong service ports or incorrect flags are enabled in the TCP header.

Once the sender's packet is packaged by the transport layer, it descends down the stack to the network layer, which is governed by IP.

D. INTERNET PROTOCOL

The network layer of the four-layer model is the IP layer. This layer is responsible for getting the packets from the source computer to the destination computer using IP addresses. IP addresses are not actual physical addresses, but instead are logical addresses used by networks to recognize the location of networks and individual hosts. The packets traverse the network by "hopping" between two hosts, whether they are computers or routers. IP addresses are assigned to each stop on the journey and this, in a nutshell, is how the packet finds its way to its final destination.

The IP address is included in the IP header which encapsulates the TCP packet which it received from the transport layer above. In addition to the IP source and destination addresses, the IP header also includes other information important to its transit such as the TTL, or time to live, what the protocol is (e.g. ICMP, TCP, or UDP), the datagram length, amplifying flags, a datagram ID number, fragmentation flag and other bits of information. For security purposes, the most important features of the IP header are the source and destination IP addresses.

In the current IP version used by the majority of systems, the IP address is 32 bits long. This equates to over 2^{32} different addresses. This number, although it seems very large, is proving too small for the high demand of IP addresses on the Internet. Therefore, a new IP version is being introduced and slowly phased in called IPv6 (for Internet Protocol version 6). With IPv6, IP addresses will be 128 bits in length, giving us a much larger address space. For the purposes of this thesis, 32 bit addresses will be discussed.

The 32-bit IP address is broken down into four numbers most often represented by four decimal numbers separated by a period (e.g. 123.59.392.41). The first section of the number gives an indication of the size of the network where the host of interest resides. The rest of the numbers distinguish between the different hosts on that given network.

The TCP/IP protocol suite does not have any built-in mechanisms to prevent a malicious user from spoofing an IP address when sending a packet. However, the mere routing behavior of the Internet backbone and the TCP/IP protocols limit the ability of malicious users setting up communication pairs with a spoofed sender’s IP [BEJT-05].

An additional IP header value of interest is the fragmentation flag. Fragmentation, or the breaking down of bigger packets into smaller packets, generally is performed by TCP; however, the IP protocol also supports fragmentation for UDP. Early signature-based IDS did not account for the fragmentation of packages and many hackers intentionally fragmented their harmful packets to fool the IDS engines into letting them on the network. As will be seen later in this thesis, the basic premise of both Therminator and Zippo addresses problems such as fragmentation.

The IP address when used in conjunction with the TCP service ports tell the system administrator a lot about a specific package. A system administrator can easily recognize glaring disparities such as a web service packet sent to an IP address that is not a web server. Often times, hackers use methods such as these to gain unauthorized access to a system. Both signature-based and anomalous-behavior intrusion detection systems use the TCP and IP headers to interpret the purpose of the packets and to gather information about their makeup, origins and destinations.

E. SUMMARY

The key to understanding how hosts communicate over a network and to understanding how many security tools, such as an IDS, work is in understanding the TCP/IP four-layer model and the TCP/IP suite of protocols. The model provides a brief snapshot of the path a packet travels from a sender to receiver as it makes it way down one stack, across the network and up the recipient’s stack. It also helps to explain the encapsulation process of the protocols contained within the model.

Transmission Control Protocol is a widely used protocol for communicating between two hosts over the Internet. It provides a connection-oriented protocol that assures proper delivery of packets to the recipient. The Internet Protocol is a set of standards that provides the architecture for providing logical addresses to networks, systems and hosts on the Internet. Additionally, it takes part in the encapsulation of packets as they traverse the Internet and get routed to appropriate destination.

The headers in both TCP and IP play a large role in IDS detection of malicious behavior and will be discussed more specifically in Chapters III, IV and V.

III. INTRUSION DETECTION SYSTEMS

A. CHAPTER OVERVIEW

This chapter will discuss what an IDS is and how it works. Additionally, it will look at the different types of IDS and the different techniques used to qualify packets. And lastly, it will provide a cursory look at some of the factors a system administrator should think about while implementing an IDS.

B. INTRUSION DETECTION SYSTEMS

Commercial IDS first appeared on the market in the late 1980’s. These were developed from products that were created mostly for government research and government networks. Most of the IDS on the market today however, evolved from products that were developed in the mid- to late 1990’s with the proliferation of the Internet [CROTH-03]. Intrusion detection is simply that. It is a tool that is used to detect if an unauthorized someone (or something) has intruded on a network. Unlike firewalls, which allow or disallow packets to travel onto the network following a rule set, an IDS is used to analyze what is in the packets or the packet headers to determine their legitimacy and to discover security violations. Often, if an IDS finds malicious packets or behavior it warns a system administrator with an alarm and it is a human that ultimately has to stop the malicious activity.

Intrusion Detection Systems come in a wide variety of shapes and sizes and the title itself covers a wide-breadth of defensive behaviors. For instance, simple log auditing can be considered intrusion detection because a system administrator can detect if unauthorized access has occurred. For the purpose of this thesis, the Intrusion Detection Systems being discussed are those systems that were developed to gather packet data on a network, analyze that data and create some kind of feedback to the system administrator, all autonomously.

Most IDS have similar architectures to include a core module and sensors. The core module generally consists of a repository for patterns or behavior heuristics, a component for analyzing the data (e.g. normalizing and comparing data against known malicious signatures or behavior) and a component to develop the feedback (e.g. alarms or graphs) for the user. Figure 2 below is a simple diagram of the elements of an IDS.

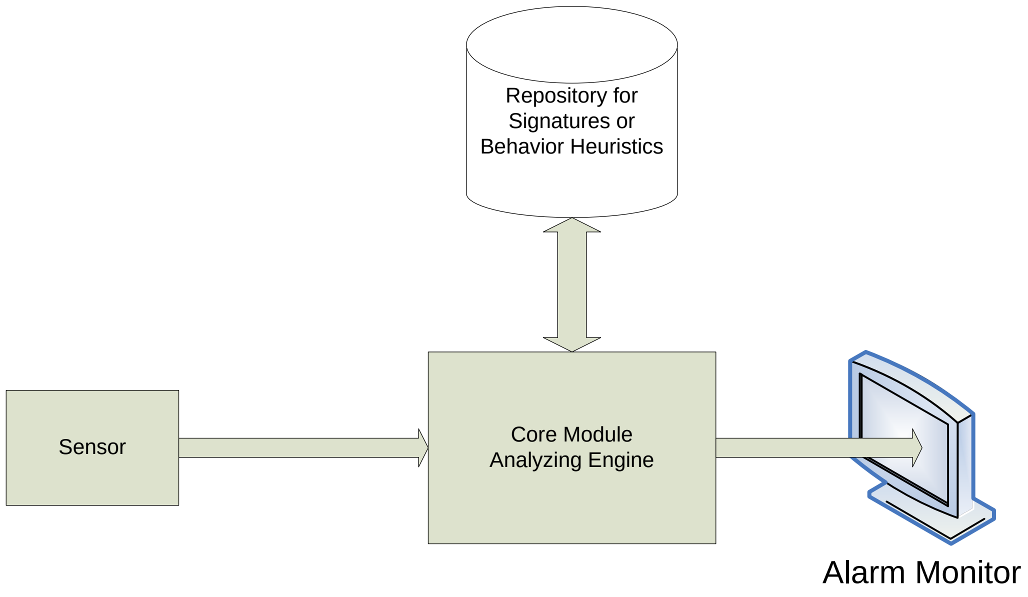
Repository for Signatures or Behavior Heuristics
Sensor
Core Module Analyzing Engine
Alarm Monitor

Figure 2. IDS Architecture Model

The alarm monitor can be a reporting system, an alarm of some type or even a human monitoring the IDS. It is simply used here as a representation of a feedback mechanism for the core module. In the core module (or analyzing engine), the data gathered from the sensor is compared with either known patterns of data or system user behavior heuristics, provided by the repository (i.e. database of malicious signatures). The sensor can be implemented in software or a hardware/software mix. It can be placed in many locations around the network depending on the information the system administrator wishes to gather. Location will be further discussed in a later section.

Most IDS can be deployed in numerous ways and can use different techniques to carry out their tasks. The specifics of Therminator’s and Zippo’s implementations will be discussed in the appropriate chapters. The following two sections discuss the general view of implementation types and techniques.

C. TYPES OF INTRUSION DETECTION SYSTEMS

Intrusion Detection Systems are available in two different types: host-based or network-based systems. What distinguishes one style from another is the placement of the sensors (or data gatherers) which determine where the data is collected. There is a hybrid that exists called network-node intrusion detection, but this is more often considered a sub-type of the network-based system [CROTH-03]. Host-based and network-based IDS each have their own advantages and their own disadvantages and sets of challenges. These will be discussed in the following sections.

1. Host-based Intrusion Detection Systems

In host-based IDS, the system runs on the monitored host. The IDS gathers its data from numerous sources on the host, to include items such as outbound packets, inbound packets, audit logs or active application processes. The primary advantage of host-based IDS is that it can see what occurs inside a given host; it gathers very specific information. Therefore, if a malicious packet (or packets) gets through the network security and takes up residence in a host machine, then the host-based IDS has the ability to see how that specific machine is affected.

Other advantages of the host-based intrusion detection system are: since the IDS records the attack at the host it can monitor what happens locally in a location where network-based IDS cannot necessarily see, there is no additional hardware needed to support the IDS since it resides on the local system and host-based IDS does not rely on signatures as heavily as network-based IDS therefore more unknown attacks can be caught.

Although there have been numerous advantages presented for host-based IDS, it is not without its drawbacks. First and foremost, a host-based IDS uses the host system’s resources, thereby decreasing performance of the monitored host. Second, if the host is compromised by a hacker, so too is the IDS. Therefore, a hacker can stop the monitoring agent and continue with their attack or they can alter the IDS reports. Because of this, intrusion detection reports taken from a compromised system should always be considered suspect. Lastly, host-based IDS generally provide an alert only after the attacker has successfully compromised the system.

The network-based IDS provides a complementary system to the host-based IDS and is presented in the next section.

2. Network-based Intrusion Detection System

Unlike the host-based IDS, the network-based IDS (NIDS) monitors and analyzes traffic as it passes on the network. NIDS make up the majority of IDS available in the commercial market. There are advantages to the network-based system that are not present in a host-based IDS. For instance, a network-based IDS sees the traffic going to all hosts on the network (if placed in the appropriate place). Because of this, network-based IDS can often facilitate stopping attacks before they can harm individual hosts. And if an attack does make it to a host, the NIDS IDS logs are not necessarily suspect as they reside on a different system than the host under attack. Next, a NIDS does not impede the performance of specific hosts on the network. Since it is placed on the network and not on a system specific, it does not require the resources of the individual systems. Because of the placement of a NIDS, it is able to analyze traffic coming from all hosts on its network collectively. This means if there is a distributed attack going on, the network-based IDS will catch it which is something a host-based IDS is not able to do. Implementing a NIDS on an entire network requires a number of sensors and at least one main analysis engine or core module. This is much less costly than installing an IDS on every machine on a network. Lastly, a NIDS will be able to see if actions other than attacks are taking place on the network (i.e. reconnaissance scans across a network to see which systems are up and running).

Even with all the positive aspects of the NIDS, it too has its disadvantages. First and very important, the NIDS has to be robust enough to handle the volume of traffic that comes across an entire network segment. This has proven to be a very difficult task for most commercial IDS considering many networks run at 100 Mbps or 1 Gbps. Second, most NIDS have very high false positive alerts. This is caused by many of the systems relying on general signatures or poorly written signatures for pattern matching and therefore many packets have patterns that correspond to these broad signatures. Many system administrators will find that their IDS alert them of many more false positives than authentic positives. Writing accurate and specific signatures with which to match patterns in malicious network traffic is a challenge for the developers of NIDS.

Network Intrusion Detection Systems assist system administrators in getting a larger picture of their network's usage as compared to host-based IDS but it does not have the ability to see what is going on at a more granular level. However, a third type of architecture exists which takes aspects of the network- and host-based IDS and combines them. This is the network-node based IDS.

3. Network-node Based Intrusion Detection System

The Network-node based IDS monitors network traffic much like the NIDS does, but the intrusion detection system is configured in such a way that different sensors monitor traffic that is intended solely for them. There usually exists one main analysis module and the sensors are distributed around a network and on specific hosts of the network (i.e. a web or e-mail server) that may be more valuable targets. The network-node based IDS combines the positive attributes of both the host- and network-based IDS.

D. INTRUSION DETECTION METHODS

In addition to the different types of systems, there are also different techniques employed to analyze the data collected on the network. The two most prominent styles on the commercial market are signature-based and anomalous behavior-based (or anomaly-based) IDS. The majority of the systems today are signature (or pattern) based IDS. A more recent style that has appeared is patternless IDS. This is the style used by Therminator and Zippo and will be discussed in Chapters IV and V.

1. Signature-based Intrusion Detection Systems

A signature-based IDS analyzes traffic with a program that compares known malicious patterns of code with the data collected from the network’s traffic. If the code in the monitored traffic matches a malicious pattern in the database, then an alert is created and perhaps either sent to the system administrator or logged in an event log maintained in the analysis engine of the IDS. There are three main techniques for signature-based IDS analysis: simple pattern matching, stateful pattern matching and protocol decode-based pattern matching [CROTH-03].

a. Simple Pattern Matching

Simple pattern matching is the most uncomplicated of the three types of signature-based IDS. Using defined patterns of code that are saved in the IDS database, simple pattern matching IDS analyze each packet on the network looking for a match to the malicious code.

b. Stateful Pattern Matching

In stateful pattern matching, the IDS has the ability to compare entire sessions with known malicious code. Instead of analyzing the data, packet by packet, the stateful IDS will examine the full session by reassembling all the packets in the conversation and attempting to match them against a known malicious pattern. This is more robust than simple pattern matching as it can catch attacks that are distributed over numerous packets.

c. Protocol Decode-based Pattern Matching

This type of signature-based IDS builds on stateful pattern matching by adding a feature to the analyzer that decodes for specific protocols as necessary. Packets are decoded at the protocol level and the session level to aid the system administrator in preventing attacks to specified protocols by focusing on broad protocol-specific rules. Often times, malicious users exploit systems by using little known aspects of protocols to gain access to the network. Protocol decode-based pattern matching assists in combating this exploitation method.

2. Anomaly-based Intrusion Detection Systems

An anomaly-based IDS analyzes network traffic with a program that compares the monitored traffic to stored normalized behavior heuristics based on network usage. Anomaly-based IDS have the advantage in detecting unknown attacks as they do not rely on defined patterns of code in known signatures. In order for this type of detection to be useful, normal behavior must be very accurately modeled to have a worthwhile comparison to use. Anomalous behavior can be as simple as a network login outside normal operating hours to a very complicated statistical pattern of use by a given individual over time (e.g. what time an individual logs in, from where an individual logs in, what type of traffic an individual generates, etc.).

Anomalous behavior detection can prove to be very unwieldy in a sizeable organization without a well established and consistent work pattern. Additionally, anomaly-based IDS are disadvantageous in that people routinely behave differently and it is difficult to normalize human behavior to such a degree that false positives are not commonplace.

E. IMPLEMENTATION FACTORS

Prior to implementing an IDS, a system administrator has many features of his network to think about as well as many different types of IDS to consider. After creating a network topology and inventory of hardware/software and services, the system administrator needs to determine what IDS architecture and data analysis will work best for his network. Additionally, the system administrator needs to determine the overall Intrusion Detection System goals. The decision can be based on the size of the network, the geography and topology, the purpose of the network (including value and services offered) and the organization’s policies and resources.

Once the system administrator ascertains the goals of the IDS and which IDS best meets them, he has to determine the number of sensors desired and the placement of these sensors, also known as monitoring zones, based on his perceived threats and where they will originate. Further, the system administrator has to decide how incidents are to be handled and how policy violations are to be dealt with.

The majority of these factors will be discussed in greater depth in the chapters that follow.

F. SUMMARY

This chapter discussed the basics of Intrusion Detection Systems and a general IDS architecture. It also presented the three main types of IDS including Host-based, Network-based and Network-node based IDS. Next the chapter illustrated the different methods of analyzing traffic against collections of data stored in the IDS, to include signature-based made up of simple pattern matching, stateful pattern matching and protocol decode-based pattern matching and anomaly-based analysis. Finally, the chapter cursorily presented some factors a system administrator should consider prior to implementing an IDS.

IV. THERMINATOR

A. CHAPTER OVERVIEW

This chapter presents an in-depth look at the Patternless Intrusion Detection System called Therminator. It offers a high level description of patternless detection and continues with the presentation of Therminator’s evolution. Next, the chapter discusses the different tools used in the configuration of Therminator and lastly it acknowledges some of the successes Therminator has realized in implementation.

B. PATTERNLESS INTRUSION DETECTION

To address incongruities in network traffic, beyond malicious patterns and anomalous behavior, a tool was conceived that bases its analyses on something wholly different than signatures and normalized human behavior, it uses statistical analysis and the principles of thermodynamics. It is Therminator.

Therminator is a Patternless Intrusion Detection System (PIDS) that uses the Ehrenfest Urn Model to categorize traffic flow within a monitored network [MARI-04]. Therminator models network traffic using categories and conversations. It bases its representation of a network on defined states and the difference in the states from one time period to the next. Therminator then interprets these states into a three-dimensional visual display using the principles of thermodynamics such as entropy, energy and temperature.

At any given moment, every network can be defined and represented by the state it is in. The changes in the state of the network are monitored and analyzed against the state boundary and decision definitions, supplied by the administrator. The visual representation of the state can quickly and accurately model what is taking place in a network at any given point in time. Therminator is a viable replacement for other IDS, but for maximum security of a network, this tool should be used in conjunction with other IDS and other tools to develop a defense in depth.

C. ORIGIN OF THERMINATOR

In 2001, two Naval Postgraduate School students, Lt. Stephen Donald, USN and Capt. Robert McMillen, USMC, developed a real-time implementation of a patternless intrusion detection system at the network operations center (NOC) in Fort Shafter, Hawaii [MCEAC-04]. This intrusion detection system was given the name Therminator and as stated above is based on statistical mechanics to translate a network into a series of states that are visualized into three dimensions by using the thermodynamic concepts of energy, entropy and temperature.

The basic algorithms for Therminator were originally conceived by Dr. Dave Ford, currently of the Naval Postgraduate School, working for the National Security Agency. Ford’s Therminator model, in essence, viewed network traffic as state transitions and modeled the system using the thermodynamic concepts [DONA-01]. Ford worked with Dr. John McEachen of the Naval Postgraduate School and graduate students, Donald and McMillen, to develop a robust version of Therminator. It has since been revised and additional functionality has been added to the latest version to include a GUI-based decision tree. This newest version was named Zippo and will be discussed in Chapter V.

D. THE BASICS

A network can be systematically studied in the same way as any other closed system. There exists in a network, traffic that has both an origin and a destination. It flows through the network just as electrons flow in a closed circuit, bits on a wire. Statistical mechanics can be employed to describe this flow of network traffic through the system. Comparatively speaking, computer science is a very juvenile discipline as opposed to physics or mathematics. If one could develop a way to apply scientific thought and behavior to a computer network, there should be a way to pattern the network traffic in such a way as to make it more understandable than an overwhelmingly vast list of IP source and destination addresses, ports, services, etc. Dr. Dave Ford did just this. He, along with others, developed a fairly complex system that simply and understandably portrays the traffic in a network.

1. Buckets and Balls

Therminator observes a network’s traffic on a microscopic level, one state at a time, and translates it into a macroscopic, easily viewed model. A three-dimensional visualization of a network is ultimately produced by the Therminator system to provide the network administrator the ability to study a real-time picture of the network’s traffic volume and what that traffic is doing. Therminator uses the concept of buckets and balls to categorize this traffic behavior. The buckets represent the conversation groups of a network and the balls represent the information being passed between the conversation groups.

The buckets are observed over pre-configured time intervals and using the thermodynamic principles previously mentioned, translated into three-dimensional graphs showing both a thermal representation of the states and the transitions between the states. The bucket and ball analogy is best described by Donald and McMillen as follows

> A bucket will contain some ordinal number of balls, and the collection of buckets and their respective number of balls will be the state of the network. The bucket can be defined using any combination of conversation characteristics including who is talking (individual hosts or networks), the language they are speaking (TCP, UDP or ICMP), or the job they are performing (client or server)…A ball is a relativistic representation of the information a bucket contains. A state transition causes a shift in the distribution of information between the buckets. This model translates network behavior into state transitions by selecting a ball from the bucket matching the source characteristics of the packet and moving that ball into the bucket matching the destination characteristics of the packet, thereby redistributing the information and transitioning the state [DONA-01].

Figure 3 below is a pictorial representation of the bucket and balls concept described above.

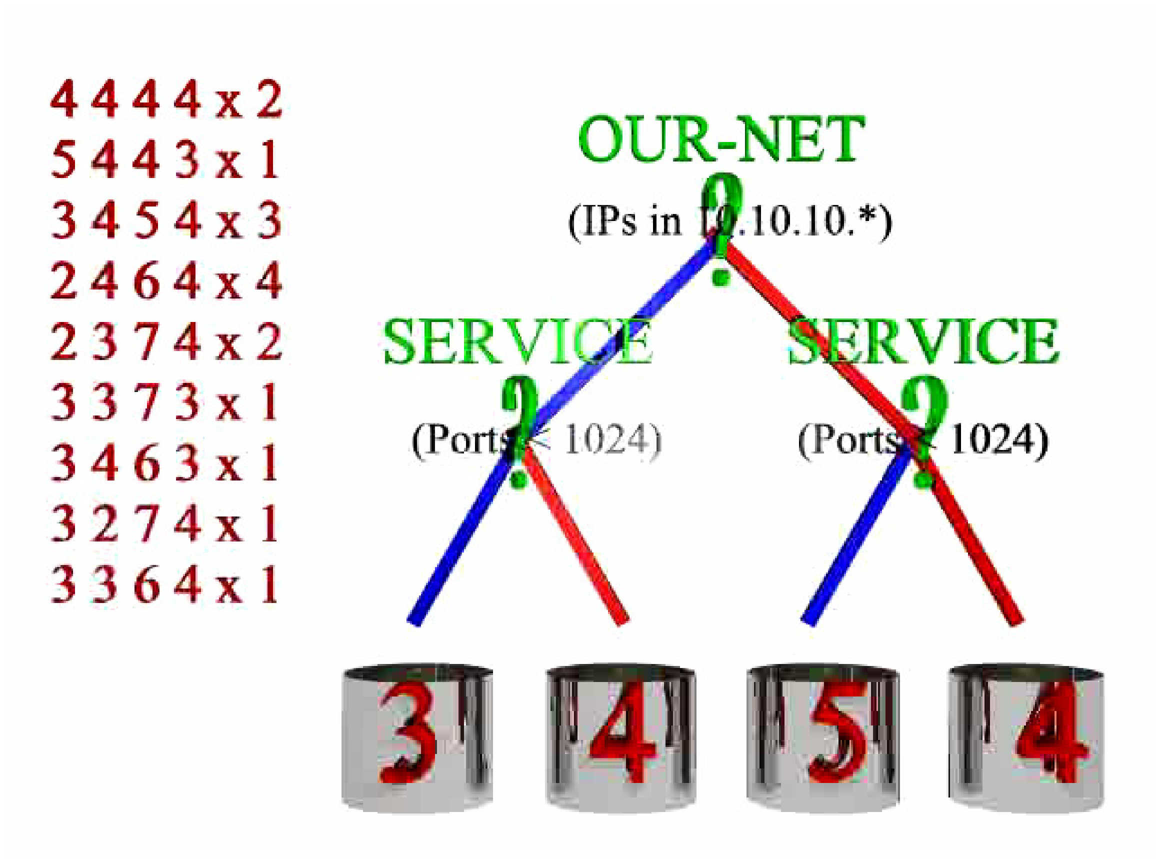

Figure 3. Representation of bucket and balls concept [From MCEATHERM-04]. The numbers to the left represent the states at given instances. The blue and red paths into the buckets are representative of the decisions defined by the decision tree.

The network state is defined by the number of balls in each and every bucket at a specified time; like a snapshot of the network's state. As packets move, the balls move. Each movement of a packet equates to a ball moving from one bucket to another (or perhaps from one bucket back into itself if it follows the same path of the decision tree). This produces a new state. The states are recorded for a defined amount of time and plotted on a three-dimensional, x-y-z axis to produce the thermal canyons and thermal towers.

2. Thermal Canyons

The Therminator GUI was developed to provide a user-friendly and intelligible visual display for the system administrator to view the traffic on his network and to monitor the health of the network with regards to network attacks, viruses, etc. Using the statistical analysis of the network traffic and translation of the state spaces, anomalies in network behavior are presented in a straightforward manner using the thermal canyons and towers graphics.

In the thermal canyon graph, the x-axis represents time, the z-axis represents the unique states and the y-axis represents the number of times the unique states occur. The z-axis is color-banded to provide the user a more visually obvious representation of any abnormalities that may occur and to differentiate between the numerous unique states. Along the y-axis, the number of times a state appears is recorded and graphed as a hill or a valley (i.e. the numbers on the axis represent the occurrences of each state represented on the z-axis).

The system administrator can see pertinent information by using the interactive features of the thermal canyon graphs. This information includes source and destination IP addresses and port numbers. Additionally, the administrator can readily view the number of balls in each bucket. Figure 4 below is a view of a thermal canyon.

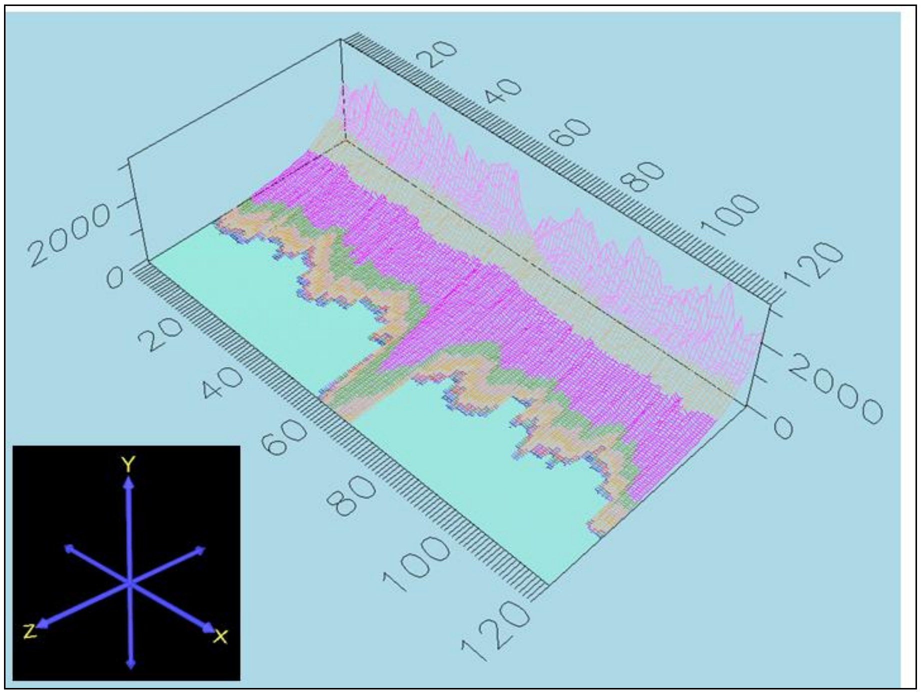

Figure 4. Thermal Canyon Visual Display [From MARI-04]

3. Thermal Towers

The other main graphics of Therminator’s GUI is the thermal towers display. The towers display, graphs time along the x-axis and average ball count along the y-axis. Each colored bar along the z-axis represents a different bucket. This graph gives the user a visual representation of the average ball count in each bucket over a defined time period. The tower heights either increase or decrease with the addition or subtraction, respectively, of information (balls) between the conversation groups (buckets). As with the thermal canyon display, IP addresses, port numbers and ball count in each bucket can be seen by using the interactive features of the graphs.

Figure 5 below is a visual representation of the thermal towers.

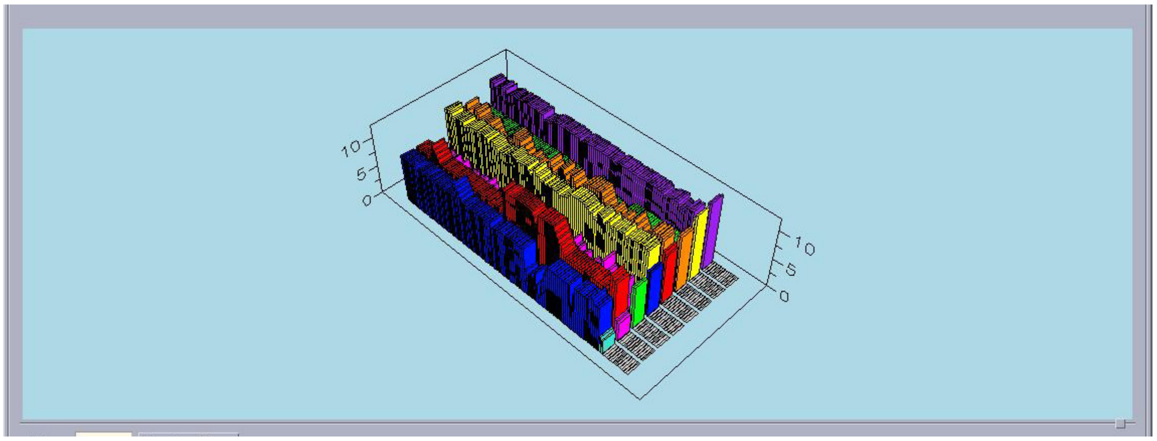

Figure 5. Thermal Towers Visual Display [After MARI-04]

E. CONFIGURING THERMINATOR

Therminator uses decision trees to categorize data into conversation groups. Some of the tools that are used to configure Therminator for a specific network include not only the decision tree, but also the slidelength and windowlength, and both initial and boundary conditions in the conversation groups.

1. Decision Tree

Therminator uses a sniffer program, also referred to as a “sucker” to gather the packet data from the network. This data, called the thermalate, provides pertinent header information to the core module such as IP address information, TCP header information and the protocol being used [DONA-01]. This data is scrutinized by Therminator and compared in a decision tree process based on rules programmed into the Therminator software. Each node of the decision tree has a rule assigned that ultimately creates the path for the balls to travel to the appropriate bucket. When a ball (i.e. packet) arrives at the node it is compared to the rules governing the node and sent down the relevant path. The following excerpt from Donald and

McMillen explains the decision tree and ball movement fairly easily

> Once, the node is located in the Matching Binary Tree, its state tables are checked to see if the packet was requested…each node has a linked list of current sessions divided into their respective protocols: ICMP, UDP and TCP. These protocol tables consist of pointers to the latest packet in a specific session or conversation with a specific host with the most recent packet pointer at the top of the list. If a packet has exceeded the user defined time to keep information in memory (MATCH_TIME), a separate thread of execution is used to clear the remainder of the state table. The length of the linked list will vary with the number of conversations or sessions [DONA-01].

2. Slidelength and Windowlength

The slidelength used in Terminator is the time defined for a single display period. The windowlength is the total period of time, defined in the code, the data is averaged over. Both are measured in seconds. Each display period on the GUI is equal to the defined time of the slidelength. For example, if slidelength were defined as four, then every four seconds data is collected and displayed for that four second period. If the windowlength is eight, then the data taken during the slidelength is averaged over the eight seconds, or two slidelengths. Figure 6 below graphically presents the concept of slidelength and windowlength.

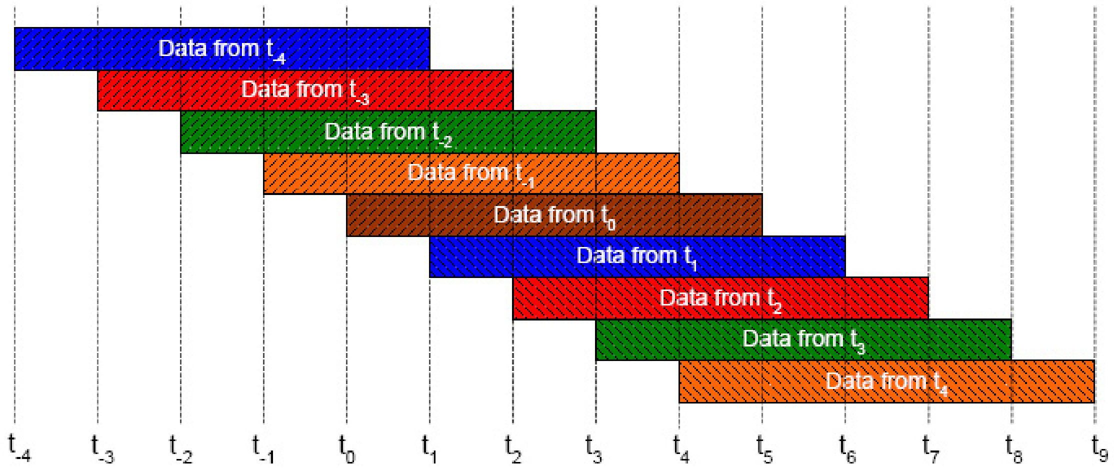

Figure 6. [From ETTL-03] The slidelength is represented in any given period between each of the time intervals. The windowlength is the period covered by a bar (e.g. period from t_0 to t_1 is one slidelength and the red bar titled Data from t_2 is a windowlength).

3. Initial and Boundary Conditions

The buckets that are defined for any given decision tree in Therminator must be assigned initial and boundary conditions. These conditions are based on the rate of packet traffic flow for a given network. Initial conditions must be set that allow the user to view a change in the space. In other words, a bucket should not be defined with an initial condition of zero or ten balls present, because if one was lost or gained, respectively, the state would not change. Additionally, using a condition close to a boundary does not allow enough variation if one or two balls are lost or gained consecutively.

The initial conditions used in the majority of Therminator experiments and implementations thus far, have put the initial ball counts in the mid-range between the boundary conditions. The system administrator has the ability to increase or decrease the number of states possible by increasing or decreasing, respectively, the initial condition.

The boundary conditions of the buckets have two limits, the upper and lower boundary. The upper boundary defines the maximum number of balls any one bucket can have. Conversely, the lower boundary defines the minimum number of allowable balls. The optimal configuration for initial and boundary conditions can be determined by referring to *Therminator: Configuring the Underlying Statistical Mechanics Model* by Naval Postgraduate Student Daniel Ettlich [ETTL-03].

F. SUCCESSES OF THERMINATOR

Therminator has been implemented in numerous government organizations. Extensive testing has been done at the Naval Postgraduate School by many graduate students since Therminator’s inception. The following sections discuss the successes of Therminator.

1. United States Pacific Command

In January of 2001, two students went to Fort Shafter at the United States Pacific Command in Hawaii to develop Therminator on an operational network. Working with a foundation based in the theory developed by Dr. Ford, Steve Donald and Rob McMillen wrote, installed and implemented a working version of Therminator. In March of that same year, Therminator achieved its first large success. Approximately 6,000 ICMP messages were seen in a four second period by Therminator. Although ICMP packets in and of themselves are not anomalous, this traffic was out of the ordinary as the owner of the client machine was logged off and at home at the time. This traffic was undetected by any of the other computer network defense monitors installed on the system [MCEAC-04].

2. Naval Postgraduate School

In May of 2001, Therminator was installed on the network at the NPS in Monterey, California. Just two months later, the Code Red virus was seen by Therminator and the system administrators were able to react quickly enough to prevent any serious damage to the network.

3. Commercial Success

In November of 2002, Atlanta-based Lancope Incorporated, a network intelligence security company, bought non-exclusive license to Therminator and implemented it in their Stealthwatch Intrusion Detection System. The high-speed data flow architecture of Stealthwatch is integrated with the data reduction and visualization technology of Therminator to provide a more robust IDS with both micro- and macro-views of a network [MCEAC-04].

G. SUMMARY

This chapter presented an overview of patternless intrusion detection as implemented by Therminator. Next, it gave an in-depth look at how Therminator analyzes the thermalate by means of a decision tree process and modeling the packets and state conversations with buckets and balls. The three-dimensional visualization including thermal towers and thermal canyons as a way of graphically presenting the status of the network was presented next as well as the coded configuration of Therminator using initial and boundary conditions of the buckets and balls and windowlength and slidelength. Finally, some successes of Therminator were presented illustrating the practical use of this patternless intrusion detection system.

THIS PAGE INTENTIONALLY LEFT BLANK

V. ZIPPO

A. CHAPTER OVERVIEW

In this Chapter, the successor to Therminator, Zippo will be presented. The motivation behind the development of Zippo, including the drawbacks of Therminator and the tasking of The University of South Carolina will be discussed. Next, a brief synopsis of the improvements to the Zippo PIDS over Therminator will be conveyed as well as the requirements defined by the Zippo development team. Concluding the chapter will be the architecture and implementation of Zippo.

B. MOTIVATION

The motivation behind the creation of Zippo, was to develop a more robust and more easily configurable program, based on the Therminator concept, that could be implemented across diverse computer systems. Zippo was also developed to address the constraints found in the Therminator system.

1. Therminator Constraints

As discussed in the previous chapter, the Therminator Patternless Intrusion Detection System has had many successes in both real-world implementation and controlled laboratory testing. However, in spite of its successes it has also had its share of constraints.

First, Therminator is not easily configured by the user or the system administrator. In order to change numerous aspects of the configuration (e.g. initial and boundary conditions and decision tree rules) it is necessary to change the computer code in the .config files, shut-down and then restart the program for the changes to take place. This is a fairly unwieldy task if the user is not familiar with the programming code of the system. Second, Therminator does not have the capability to be used in a distributed fashion across a computer network. In Therminator, a single sensor is used to gather the thermalate and then report back to the analyzing engine. This severely limits the amount of network traffic data that Therminator can collect. Third, the three-dimensional graphic visualizations known as the thermal canyons and thermal towers in Therminator are created using an expensive commercially licensed product that can only be used on the Solaris 8 system. These three main constraints prompted further research into improving the next generation of the Therminator, Zippo.

2. University of South Carolina Tasking

Dr. John Zachary and his research team at the University of South Carolina were tasked by the Applied Research Laboratory at Pennsylvania State University to develop a more robust software implementation of the network anomaly detection model that is the basis for Therminator. The new system needed to be modular, scalable and configurable for diverse computer systems [ZACH-04].

Dr. Zachary and graduate student Jun Ling developed Zippo in response to this tasking. It has been installed and tested and proves a worthy adversary of anomalous behavior on a network.

C. ZIPPO OVERVIEW

Dr. Zachary and his team developed Zippo, a system based on the Therminator concept that simplifies and reduces the amount of code necessary to implement a more robust, distributable and modular Patternless Intrusion Detection System. In addition, it has an extremely user-friendly GUI that adds significant ease in the use and configuration of the system.

In the original version of Therminator, the user had to physically change the code in the .config files to define the rules that make up the decision tree and to identify the initial and boundary conditions of the conversation groups. The new java-based GUI, provides the user with an interactive interface where decision trees are created by dragging and dropping branches and rules are defined by means of interactive text boxes. Additionally, the initial and boundary conditions of the conversations, or buckets, are created interactively with text boxes as well.

Zippo has also enhanced the visual displays of the thermal canyons and thermal towers by creating a three-dimensional interactive “fly-over” feature. This user can use the mouse to rotate the visual display to view it from any angle as well as zoom in and out for greater clarity of a given time period. The feature still exists that allows the user to “drill down” to specific buckets and or time periods to see exact volumetric amounts.

Another added feature of Zippo is the ability to define several decision trees for a single network and the ability to open up thermal canyons and thermal towers for each of the decision trees. This allows the user to see many different aspects of the network at the same time. This can allow specialized configuration of Zippo for networks that have a disproportionate amount of other-than-routine network traffic or individual monitoring of each service offered on the network. Moreover, Zippo was further developed so that numerous sensors could be placed on a network and monitored individually or by using a single decision tree with a single graphic visual representation of the thermalate gathered by all sensors and collated in one report.

D. REQUIREMENTS

Zippo was developed with the following requirements:

- Therminator's basic modeling and interaction functions are to remain the same.
- The design is based on generally accepted design architecture, is open and well-documented
- Implementation is easily modified and expanded.
- The architecture of Zippo is created using technologies that are readily available and without troublesome commercial licenses.
- Zippo can be easily installed across diverse computer system platforms [ZACH-04].

Additional requirements for Zippo that were noted in *Zippo: A Robust and Portable Network Anomaly Detection System* [ZACH-04] include a graceful shutdown and the ability to analyze and process a realistic flow of traffic at an industry standard speed.

E. ARCHITECTURE AND IMPLEMENTATION

Zippo’s foundational architecture is a Model-View-Controller (MVC) representation that is often used for GUI-based software systems [ZACH-04]. The functions carried out by Zippo, including the collection of thermalate, the analysis of the thermalate, and the visual representation of the network traffic can be mapped onto the MVC architecture as follows:

- Model: responsible for maintaining the state space model and carrying out the computations that lead to the thermodynamic representation of the network traffic
- View: responsible for presenting the visual imagery of the thermal canyons and thermal towers to the user. Also responsible for presenting the interactive features of the GUI.
- Controller: responsible for translating the traffic packet data into thermalate, implementing the user’s interactive input for configuring Zippo and stores the thermalate

1. Model

The model is analogous to the core module/analyzing engine that was presented in Chapter III. The analysis and computations of the thermalate are applied in this segment of the MVC architecture. This is, in essence, the heart of they Zippo system. It is the most memory and processor intensive of the three components in the MVC architecture. The features in the model component are implemented using Java code.

2. View

As stated above, the view component realizes three-dimensional visualizations of the analyzed thermalate in the form of thermal canyons and thermal towers that were discussed in Chapter III. The view also entails the GUI that pertains to the Zippo Control Center (ZCC) that will be described in the next section. This component ensures a proper presentation to the user and gathers the interactive data inputted by the user to be sent to the controller for translation into Zippo configurations. The GUI, like the model, is implemented using Java code.

3. Controller

The controller is the conductor of the Zippo system. It processes the network packets obtained by the sensors, into thermalate, it stores these packets for traceability and forensics data and it configures Zippo according to the data entered by the user via the ZCC GUI. The ZCC is the key module of the Zippo system that accomplishes the actual configuration. The following tasks can be configured using the ZCC [ZACH-04]:

- Network sensor and core configurations
- Decision tree construction and modification
- Administration of the thermal canyons and thermal towers
- Storage of the thermalate

The ZCC is implemented using Java code and the sensors are implemented with C++ for remote sensors and with Java code for local sensors.

4. Application Components

To implement the gathering, analysis and visualization of network traffic data, Zippo uses three components: the sensors, the analysis engine and the GUI. This follows the architectural model described in Chapter III. The sensors can be either local or remote sensors. Once they gather the network traffic data and translate it into thermalate, they pass the information to the core via a secure socket in OpenSSL. The sensors rely on OpenSSL as just mentioned as well as pcap libraries. These are open-source code libraries that are readily available to everyone. After the thermalate is received by the analysis engine, it stores a copy of the packet information in a database and processes the thermalate based on the decision tree rules and bucket conditions defined by the users. The information deduced by the core is then sent to the GUI, via TCP sockets, to fabricate the visualization graphics for viewing by the users. All three components of the Zippo application can be installed on different network components and can communicate via the network using the protocols previously mentioned.

An improvement of Zippo, over Therminator, as mentioned previously is the ability to use distributed sensors for data gathering as well as the capability of viewing numerous thermal canyons and thermal towers for different decision tree rule sets. This allows the system administrator a great deal of leeway in configuring the application for his specific system and traffic.

F. SUMMARY

This chapter introduced Zippo, the next generation of Therminator type PIDS. It presented the motivation behind Zippo, including the constraints of Therminator and the tasking by Pennsylvania State University Applied Research laboratory and presented an initial overview of the application. Next, it outlined the requirements that were generated prior to the development of Zippo. And finally, the chapter explained in detail the architecture and components that make up the Zippo application.

VI. NETWORK TOPOLOGY

A. CHAPTER OVERVIEW

In this chapter, the first step of implementing an Intrusion Detection System is investigated. It focuses on the network’s topology, its component inventory, offered services and monitoring zones. Next, it looks at the network design and discusses some parameters that system administrators should take into consideration when creating, or rearranging, their networks. Finally, this chapter addresses the topology’s relevance to IDS.

B. NETWORK INVENTORY

The first and most important step in securing a network is recognizing what network assets exist and which of them are the most valuable. Time and time again when interviewing system administrator experts, this point was made. Without knowing what valuable assets an organization has, how can a system administrator be expected to protect them? The first step in discovering the most valuable assets, is to create a representation of the network topology; an extremely detailed physical map of the network.

1. Topology

A network topology should include all hardware, interfaces, IP addresses and services offered in a comprehensive and accurately constructed map of the network. Internal and external connections should be annotated and perimeters should be clearly marked. Security systems such as firewalls and intrusion detection systems should be depicted in their specific locations, to include the position of the IDS sensors. Additionally, demilitarized zones (DMZ), or areas outside the perimeter often guarded by firewalls and made up of common Internet accessed servers such as web and email servers, should be illustrated.

For large networks, a system administrator can create a suite of topologies that start at the highest level and progressively focus in on more specific assets such as individual monitoring zones. A monitoring zone is a section of the network where the assets share certain privileges based on their trust levels [BEJT-05]. One example is the perimeter of the network. The perimeter is often one of the most vulnerable interfaces of a network. Its protection is crucial in that “outsiders” can gain access from the Internet to a network via the perimeter.

Once the logical map is created for the physical network, it should be considered sensitive information and not readily shared with individuals who do not have the “need to know” the network’s design. In the wrong hands, a network topology can create a treasure map for malicious users to find their way in.

Once the map is completed, the system administrator should create an itemized inventory of all the network’s assets including serial numbers, physical locations and operating systems/software. Additionally, the relativistic importance or value of the assets should be annotated so the administrator can ensure the organization’s most important systems are given their due attention.

2. Services

In conjunction with knowing the physical and logical layout of a network, it is also important to know what services are offered by the systems in a network. As discussed in Chapter II, network services can generally be cross-referenced to TCP ports on computer systems. Services that are allowed in specific hosts can be defined by the ports they can be accessed by and this bit of information is essential to an informed system administrator. If a port is open on a host and the host can be reached from outside the perimeter, then hackers can access the host if it is not safeguarded. If a system administrator is unaware of the services that his hosts offer, he will be greatly challenged to keep those hosts protected from either internal or external threats. A compiled list of hosts and their services is a critical tool for the system administrator.

3. Threats

Just as important to a system administrator as knowing his assets and his network’s active services, is his knowledge of the origin of his network’s threats. A system administrator should know who or what he is protecting against. This helps when developing the network design, as it provides additional information with which to create an architecture that addresses the threats.

C. NETWORK DESIGN

In the creation of a network, or development of its topology, it is important for the system administrator to focus not only on the functionality of the network but also on an architecture that will emphasize security.

There are numerous security factors to take into consideration when creating a network design. First, the network should be designed with the intention of making access to unauthorized individuals as difficult as possible, while still providing as much functionality as necessary for the organization. This means not only examining access from outside the network, but controlling access and maneuverability inside the network in case of a breach of the perimeter or internal malicious users. At the same time, the design should make certain that the network is not so complicated that those monitoring its security cannot observe what is taking place within.

Another factor in creating a secure network design is ensuring that access points, available services, active ports and connections are kept to as minimal level as possible for desired functionality and compulsory redundancy. Services that are automatically offered by operating systems should be turned off unless they are absolutely necessary. Additionally, safety measures and policies should exist that prevent the system users from turning services on without the appropriate authorization and privileges.

Taking the above factors into consideration improves a system administrator’s chances at keeping his network secure, however, beyond the design and configuration of the network and its hosts, the system administrator must also create his network to facilitate ease of maintenance. Every week, patches and updates are provided to information technology users to fix bugs and weaknesses in their systems so malicious users cannot exploit them to gain unauthorized access. When designing a network, an administrator needs to make certain that the architecture of the network is such that he has unobstructed access to the systems in order to apply the patches and updates. The entire network needs to be easily updateable. If even one host on a network is not kept current, an entire system can be exploited.

The suggestions for network design above are obviously not all-inclusive; however, they provide sound advice to system administrators developing network architectures and address several issues that can easily be overlooked if network functionality alone is focused on.

D. IDS

When implementing an IDS, it is critical that the system administrator has total knowledge of the network in question. Asking an administrator to secure a network with no, or limited knowledge of its topology, interfaces, access points and connections is like asking someone to drive across the country, blind-folded and without a guide. In other words, it is impossible.

As discussed in previous chapters the PIDS, Zippo, relies on sensor placement and knowledge of network configuration with respect to IP addresses and services, to create a worthwhile visualization of the network’s traffic. If a system administrator is ignorant to the location of the network’s servers, the services they offered or the authorized users, he would not be able to configure the decision trees to provide him any information that could assist in the network’s security. Zippo would be useless to him. Therefore it is important for the administrator to know his network intimately.

E. SUMMARY

This chapter presented the most important aspect of creating a secure network, knowing the network. It detailed the important steps of creating a network topology, knowing the services and knowing the threats against the network. It presented factors to keep in mind when creating a network design and ultimately offered the importance of intimately knowing the network to implement an IDS.

VII. ORGANIZATIONAL POLICY

A. CHAPTER OVERVIEW

In this chapter, the elements of good policy development will be presented, along with sources of policy creation. In addition, factors that influence organizational policy will be looked at. Lastly, security elements that implement policy will be presented as well as thoughts on publishing and distribution of policies.

B. POLICY CREATION

An important asset that every organization should have to assist in securing its network is an organizational policy pertaining to network use and security practices. A thorough and unambiguous policy provides a system administrator with a tool that he can use to make control and access decisions. Having a policy approved by an organization’s executives and ensuring it is enforceable is crucial to a system administrator attempting to defend a network. Policy is implemented by security tools such as firewalls and IDS. Rules applied to these security tools are defined by the policies of the organization. For example, prior to implementing an IDS on a network, the administrator must know what the policy is in order to configure the IDS to recognize normal traffic and anomalous behavior. But what makes a good policy?

1. Elements

A policy has three main elements: authority, scope and expiration [NORETAL-03]. The authority of a policy refers to its status in the hierarchy of rules that regulate the network’s security. In other words, where does the policy’s authority originate. For example, in the Department of Defense, there are numerous levels of authority that apply to organizational policies. Generally, these are hierarchical. Organizations that create their own policies must adhere to the guiding principles of those commands superior to them in their chain of command. Policies created at subordinate commands can be more restrictive, but not more lax than those which govern them. The authority of a policy is closely related to the scope.

The scope of the policy refers to who the policy applies to. Does it apply to the entire organization, or sub-groups within the organization? Does it apply to just the perimeter or the internal network as well? This is a very important aspect of creating an unambiguous policy and generating the rules that enforce the policy. When implementing intrusion detection systems, if the scope of the policy is unknown, effective rules cannot be written. And since networks, organizations and threats change, so to should the scope of policy.

An organization’s IT security policy should be reviewed and updated on a regular basis as well as when specific circumstances dictate. The scope and the authority that prescribe rules and regulations for the policy are not static. Neither are the threats that imperil a network’s safety. All of these elements, authority, scope and expiration must be taken into consideration when creating an organizational policy.

2. Written

Most organization’s have more than one written policy that addresses IT security. It is important for the system administrator to thoroughly research all policies prior to implementing an IDS. Better yet, it would behoove the administrator to create a single policy containing all the rules pertinent to the network. It is much easier to create control and access rules from one compiled document than from numerous documents floating around an organization.

Written policy can be found in more than just IT documents. It can be found in organizational memos, employee handbooks, directives from system administration and many other publications. It is the system administrator’s challenge to create rule sets that address all the policies, not to mention de-conflict the policies to be enforceable and practicable.

Written policies are a way of communicating the rules and regulations that govern the use of a network to all users. The system administrators need to make certain that the policies correctly reflect the desired allowable uses of the network. Since the control and access of the network is based on the policies, it is imperative that these rules and regulations clearly reflect allowable and unallowable behavior.

3. Verbal

Along with the written publications in an organization that dictate policies, verbal guidance also exists. This can be formal or informal; although it generally tends toward informal (e.g. “the administrator said I could”). It is important for the system administrator to take the verbal policies into consideration when creating the overarching guidance for the organization.

Verbal policies should be analyzed to determine if they are in adherence with published policies and best practices of network security. All too often, weaknesses are created in secure networks because users download unapproved programs or enable unauthorized services as they are not made aware of the network’s security policy or the governing rules and regulations. Instead, they are ignorant of the repercussions of informal or non-existent policies.

Verbal policies should be discouraged and the rules and regulations that govern an organization’s network ought to be formalized, written and approved by an organization’s authorities. If there is an absolute need to change network policy “on the fly”, the changes should be documented and evaluated against the standing policy to ensure gaps are not created in the network’s security.

C. POLICY INFLUENCES

Policy can be influenced by any number of things. People, hardware, network topology, threats, etc. are all factors that should be taken into consideration when developing policies. The following paragraphs address just a few of the important influences on network policy.

Network components’ physical locations and logical connections are very important influences on the development of network policy. First, from a physical standpoint, hardware, software and often times knowledge residing on systems, must be physically secured by locks, security guards or other means. Policy needs to reflect the physical security of a network.

In addition, the logical topology of the network dictates policy creation. The network perimeter, the DMZ and the servers that allow public or remote access are all factors that should influence the security policy. Creating a highly detailed network map and inventory as discussed in Chapter VI is very important for establishing the policy’s foundation. If the system administrator does not know what his assets are, where they are located, their IP addresses, installed applications or what services they are running, he cannot create rules that accurately focus on the threats.

Along with the physical assets and the network topology, the threats to a system also heavily influence policy. The administrator needs to make certain he knows what assets are high-value targets and where the threats originate. Is the Internet a concern? Is malicious internal traffic probable? Knowing what the threats are enables the administrator to properly address them with policy.

Lastly, but definitely not least of all, people influence policy. Users directly influence policies from the standpoint of identification, authentication, privileges, authorizations and unintentional misuse. Policies need to address all of Murphy’s Laws in network administration. In developing policy, a system administrator should assume that anything a user can do wrong will be done and he should create his policies to address that concern.

D. POLICY IMPLEMENTATION

Policy is implemented in numerous ways. Physical security directly implements the physical safety of network assets. Firewall rules are applied at the perimeter to carry out policy that allows/disallows specified network traffic. Rules of behavior and regulations of network use are published and are guides for network users to follow in order to adhere to an organization’s policy. Management implements policy by holding individuals accountable for their actions and for observing the policy rules.

What is important for system administrators and the management of an organization is to ensure that policy that is implemented is supported and put into practice by all users. Policy should address the goals system administrators have for their networks. It should be implemented to support the network’s purpose and the organization’s mission. However, above all else, implemented policy must ensure safety to the best of its ability.

The security plan, presented in the next chapter, is an ideal tool to implement an organization’s policy. Its sections include thorough and detailed information pertaining to the security of a network. Chapter VIII presents the security plan in fine detail and will further discuss how policy is implemented.

In order to implement policy to users, it is important that the policy is published and distributed accordingly.

Generally an organization will have different levels of users and the policies should reflect the privileges afforded to each. Similar to the network inventory and topology, policies concerning firewalls, IDS, system monitoring etc, should be held as sensitive documents. If malicious users can gain insight into the rules that protect a network, they can exploit the holes created by them.

E. SUMMARY

This chapter presented the creation of organizational policy with respect to networks. It discussed the elements that make a good policy and policy creation based on written documents and verbal understandings. Some of the influences of policy creation were presented as well as different tools that assist in implementing the policy.

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. SECURITY PLAN

A. CHAPTER OVERVIEW

This chapter presents yet another asset that should be in place prior to implementing an IDS. The purpose of a security plan is presented as well as the components that make up a security plan. Lastly, guidance and references will be included for the system administrator's development of the security plan.

B. PURPOSE

What is a security plan and why is it needed to help keep a network secure? According to many government and commercial organizations, a security plan is a thorough document that touches on every facet of a network's security and compiles it neatly into a "one-stop shop" for finding security requirements, controls, policy, responsibilities and future plans to meet any shortcomings determined in the development of the plan. In fact, most government agencies are required by the Office of Management and Budget (OMB) Circular A-130 and Public Law 100-235 to develop and maintain a security plan [NIST-98].

A security plan is developed to protect an organization's assets. It documents every facet of an information technology program to include not only those aspects listed in the above paragraph but also such items as responsibilities and expected behaviors of the all individuals who access the system, physical security requirements and managerial, operational and technical controls to address the appropriate risks. Without a security plan an organization would have an uneven and ad hoc approach to network security, and weaknesses would be much more likely, if not absolute.

It is very important that a security plan is developed for a network. As stated numerous times before, adding security measures in an unplanned or informal manner can cause just as much harm to a network as it can provide protection. A security plan is a required asset that should be in place before Intrusion Detection Systems are installed as the plan provides significant information and details for creating the IDS rules.

The system administrator is responsible for ensuring the security plan is developed. The system owner is ultimately responsible for the plan, but for purposes of this thesis, the administrator is considered the owner. Besides developing it, the administrator is also responsible for implementing and maintaining the plan. It should be considered a living document and should be revered as the law in managing the network.

When the plan is completed it will contain technical information about the network, security requirements of the network and the controls that are or will be put in place to address the risks and vulnerabilities of the system. Because of this, the security plan, just like the network topology map, the asset inventory and the policies, should be treated as a sensitive document and only those sections that are meant for the public should be distributed. If a malicious user were to obtain any of the documents presented thus far, he would have enough information to exploit an organization’s network.

For purposes of this thesis, the NIST Special Publication 800-18 *Guide for Developing Security Plans for Information Technology Systems* is used as guidance.

C. COMPONENTS

The following subsections outline the components that should be included in a thorough security plan: General Information, Management Controls, Operational Controls and Technical Controls. This is a brief presentation of them and more detailed information can be found in the NIST Special Publication 800-18. The components listed below follow the NIST publication's outline [NIST-98].

1. General Information

The first section should include general information about the system in question. It should contain information regarding who is responsible for the system (i.e. organization and personnel) and who maintains the system to include contact information. If there are different personnel assigned to other functions such as a separate security officer for the system, these people should also be listed with their contact information.

The system operational status must be defined for the system. It can include either operational, under development or undergoing a major modification. If different monitoring zones in the system fall under different statuses, then all zones and their status should be reported. Along with the operational status, a description and purpose of the system and the operational environment of the system must be presented.

The last section under the general information segment should include the system interconnection information, which includes the connections used by the system for sharing information resources, and a description of the types of information handled by the system and its criticality. The laws, regulations and policies pertaining to the confidentiality, integrity and availability of the data in the system should be presented here as well as a description of the data’s sensitivity.

2. Management Controls

In this section, the management controls that are either in place or that will be put in place are presented. Management controls are those controls that address the management of the system, the management of the information contained in the system or the management of the system’s risk. To determine the controls that need to be put in place, a risk assessment should be done. This should include determining the value of the system’s assets (including the data contained within), threats to the system, vulnerabilities of the system and the safeguards currently in place and their effectiveness.

The OMB Circular A-130 requires that all systems receive an independent review of their system on a periodic basis. The management controls section should contain information pertaining to the review including the type of review conducted and any of the findings. Additionally, it should address the planned corrections of any deficiencies found during the review.

Rules of behavior are presented in the management controls section as well. These rules should address expected behavior of all users, responsibility delineation of users and administrators, limits on interconnection and restoration priorities in case of failure and other topics such as remote access, accountability, use of equipment, etc. Additionally, the rules of behavior should include the penalties of violating the rules and should be provided to all users and acknowledged (in writing) prior to system access being granted. Different levels of behavior may be necessary for different systems. System administrators should have behavior rules evenly balanced with their responsibilities. For this reason, system users and system administrators may have totally different sets of rules and the management control section should address both.

The rules of behavior should be written in line with the policies of the security systems. If a firewall is setup to disallow ftp traffic, then the administrative rules should state that ftp is not allowed. This way, there is redundant system of checks.

The last part of the management controls section consists of the life cycle security of the system. Here, the plan should include how security is implemented in each phase of the system's life cycle. Special Publication 800-18 defines five phases to a system's life cycle that must be addressed: Initiation, Development or Acquisition, Implementation, Operations and Maintenance and Disposal.

3. Operational Controls

Operational controls are those controls which address the security measures implemented by people, vice policies implemented by the system itself. Controls that should be addressed in this section include: Personnel Security, Physical and Environmental Protection, Production and Input/Output Controls, Contingency Planning, Application Software Maintenance Controls, Data Integrity/Validation Checks, Documentation, Security Awareness and Training.

The operational controls section focuses on the human factor of network security. The personnel security, physical and environmental protection focus on the safekeeping of the system assets from malicious users and security attributes of the physical location of the system, to include such problems as fire safety and pilfering. Production and input/output controls deal with security steps for proper handling, copying, transferring, inputting, printing, storing and disposing of data contained on the network, as well as, the maintenance of application software.

Contingency planning, discussed in the operational controls section, entails a description of the steps carried out to ensure backups exist in case of a contingency. This should include frequency of backups, location of backups, roles and responsibilities of users and administrators in case of contingency and should annotate if any additional disaster or contingency plans exist for the organization and where to find them.

The data integrity and validation controls subsection should include information on any virus software, integrity checkers, firewalls, IDS or other security systems in place to maintain and check for data integrity and validity. This section contains policies such as how often passwords are checked and if message authentication is used. In short, this subsection covers any application or control that is used to ensure the integrity, authenticity and validity of data on the network.

Next, the operational controls section addresses two very important administrative actions. The first is documentation. This subsection should list all documentation that pertains to any hardware, software, policies, rules and regulations, backup activities or any other facet of the network that has documentation. Maintenance of documentation is very important for a system administrator to preserve some semblance of order on his network. Too often applications, policies or configurations are changed without being documented and without a system administrator’s knowledge and this is a detriment to the security of a system. Documentation should be thorough and readily available to those who need it.

The second administrative feature presented in the operational controls section is security awareness and training. This is a very important and necessary control to implement. A system is only as secure as the individuals using it. The security plan should reflect what training is provided, how often, to whom and how the responsible training personnel ensure everyone is adequately trained. This section should not be taken lightly as many viruses, back doors and other malicious behavior is initiated by unsuspecting and unknowledgeable system users.

Lastly in the operational controls section, is the capability of the organization to provide incident response. Here, the organization should publish what steps are to be taken in the case of a breach of the network’s security. The incident handling procedures should include such items as reporting requirements, responder’s actions and what preventive measures are in place. Carrying out a well-rehearsed incident response plan is important to minimizing the amount of damage an intruder or an attack can inflict.

4. Technical Controls

The technical controls of a system are those that are executed by the system itself. They may be initiated by a user, but they are carried out automatically by the system. The first of these controls presented is for identification and authentication of users. The security plan should describe how users are uniquely identified and authenticated. The techniques and mechanisms used to carry out the identification and authentication process should be explained in detail here.

Once the identification and authentication processes are detailed, then the authorization and access controls can be presented. These controls delineate who is allowed to access what. This section should explicate how the system ensures only authorized personnel gain access to the allowable systems or applications as well as the organizational policies that define what authority is granted to each of the users. If encryption is used by the system, this section should address the methodologies utilized. In addition to the logical access controls for the organization’s personnel, public access controls should be defined for applications that are accessible by the general public.

The last component addressed in the technical controls section is the audit trail. This is a record of system or application use by specific users or processes. An audit trail can be very helpful in intrusion detection, user accountability, system health and other facets of network administration. This section should discuss the audit trail mechanisms in place on the network.

D. GUIDANCE

As stated above, the NIST Special Publication 800-18 is a very thorough document pertaining to security plans. It is simply a guide to aid the system administrator in development of the security plan. Other NIST publications as well as other government documents are available to assist with the composition of the security plan.

System administrators need not reinvent the wheel when creating their security plans. Many templates and completed plans are available from commercial and government organizations.

E. SUMMARY

This chapter introduced the NIST Special Publication 800-18 Security Plan. It discussed the importance of having a security plan in place to defend a network and to provide adequate documentation and guidelines to system administrators as well as system users. The components of the security plan were discussed and were broken down into sections addressing managerial, operational and technical controls that should be established for the network. Lastly, a brief section discussing general guidance was provided.

THIS PAGE INTENTIONALLY LEFT BLANK

IX. CONFIGURING ZIPPO

A. CHAPTER OVERVIEW

This chapter presents the methodology to implement Zippo on a system. It starts with a brief explanation of the threats and vulnerabilities of a system as they pertain to Zippo. Next it discusses the hardware requirements and configurations of Zippo and the optimal sensor placement. Lastly, it presents the methodology for configuring the decision trees (buckets and balls).

B. RISKS AND VULNERABILITIES

As discussed in the previous chapters of this thesis, the most important piece of information a system administrator needs to protect his network is what specifically he needs to protect and from whom or what he needs to protect it. Carrying out a risk assessment as part of the security plan and creating a network topology and inventory are key steps in finding the high value targets of an organization’s network and the vulnerabilities that exist within a network. If an organization’s lifeblood is the data kept in a data warehouse contained in their network, then their primary security concern should be keeping that data warehouse safe and impenetrable to unauthorized users.

Most organizations have high-value targets. Whether they are as simple as files that contain personnel information or as complex as global data warehouses, organizations need to protect them. The completion of an organizational policy, security plan and network topology and inventory is vital to a system administrator in determining what his high value targets are, where the threats to these targets lie and the vulnerabilities on the network that may exist.

Once the administrator knows what he has to protect, he then must determine his overall objective for Zippo. Does he want to employ Zippo as a simple monitoring device to show the traffic on the entire network or just the traffic intended for one service? Does he want to create an interface that permits an alert program to “read” Zippo’s visual displays and alert him when the traffic is out of the ordinary? To determine these objectives, the system administrator needs to equate the abilities of Zippo with his network’s risks and vulnerabilities.

Zippo can assist the system administrator in detecting attacks that threaten specific systems, attacks that get past the firewall and attacks meant for specific services. Additionally, Zippo can also provide the system administrator with important details about normal network usage and specific information regarding the source, target and types of attack that happen on the network.

Once the system administrator knows his objectives, his risks and his vulnerabilities he is ready to deploy Zippo on his network.

C. HARDWARE AND SOFTWARE

As discussed in Chapter V, Zippo was not developed as a platform specific application. Its sensors and analyzing engine are written in Java code and C++ and can be deployed on different hosts throughout the network or on a single host. There are software requirements to properly run

Zippo, but all programs are open source and can be downloaded from their Internet sites. The sensors rely on Open SSL (libssl and libcrypto) and pcap (libpcap) libraries [ZACH-04]. The core, written in Java code, requires Java Run Time to properly execute. It was developed using Java 2 SKD 1.4.2, so the developers recommend this version to properly employ Zippo.

Hardware availabilities should be considered prior to deploying Zippo on a network. Of course, the ultimate factor in configuring systems is resources, but if the resources are available the following presents optimal configurations.

The core function of Zippo requires a substantial amount of processing power, as does the visual display graphics presented by the Zippo Control Center. The more CPU power, RAM and hard drive storage space available, the more capable and efficient Zippo will be. A beneficial hardware configuration for Zippo is a dedicated host employed specifically for its core functions and a dedicated host for its GUI component. Processing and storing millions of packets of thermalate requires a lot of CPU horsepower and isolating Zippo on its own host keeps it from having to “fight” with other applications for a share of the processor. Additionally, the GUI component of Zippo requires dedicated processing power for real-time three-dimensional visualization of the thermal canyons and thermal towers and to run the ZCC which allows the user to interface with the interactive GUI in constructing Zippo’s many configurable variables. The user needs to make certain that the GUI component is connected to the core process using the IP address and TCP socket of the system running the core component.

The sensors of Zippo can be realized in either a remote host or on the local host of the core component. Zippo can have only one sensor or many sensors can be deployed in a distributed manner. Placement of sensors will be discussed in the next section. Remote sensors are coded in C++ and local sensors in Java code. As stated above, the sensors rely on OpenSSL and pcap libraries, therefore these source codes must exist on the host(s) where the sensor(s) reside. Like the GUI component, the sensor’s configuration file must stipulate the IP address and TCP port of the core component as well.

The hardware used for sensor hosting is dependent on the amount of traffic flow that needs to be sucked off the network for transforming to thermalate. In Table 2 below, recommended sensor hardware is presented.

Component	**Lightly Used T-1 or Less**	**Well-Used T-1 to Lightly Used T-3**	**Well-Used T-3 or Higher**
CPU	Pentium II 300MHz	Pentium III 750MHz	Pentium IV 1GHz or more
RAM	256MB	512MB	1GB or more
Hard Drive	20GB	80GB	240GB or more
PCI bus	32 bit	32 or 64 bit	PCI-X or PCI Express

Table 2. [From BEJT-05] Hardware recommendations for sensors of Intrusion Detection System

D. SENSOR PLACEMENT

To optimally place the Zippo sensors, the system administrator must make use of his knowledge of the system’s assets, threats, vulnerabilities and offered services. The sensor placement should address the policies of the organization and the vulnerabilities discovered in the network’s topology. They should be situated in the network such that the traffic of interest can be seen by them for forwarding to the core component. Additionally, the sensors should be adequately protected so malicious users cannot access and exploit them.

1. Valuable Assets

In creating the network topology and inventory, policies and security plan, the system administrator has collected all the information he needs to appropriately set his sensors. His primary focus should be on the critical systems and the network weaknesses. The sensors should be able to see all traffic destined for and coming from the critical systems.

If a network is connected to the Internet (or other interfaces for sharing information), this is a vulnerability that must be addressed by Zippo. A sensor would be well placed to see all Internet traffic destined for the organization’s network and leaving it. If this proves to be too weighty of a task for the sensor due to the volume of traffic on the network, numerous sensors can be placed to address the different subnets of a network. Additionally, a sensor should be placed inside the firewall of the organization’s internal network to process incoming and outgoing traffic that has gotten past the firewall or originated internally, respectively.

If a network contains a DMZ which allows public access to numerous servers (i.e. Web servers, DNS, etc), a Zippo sensor should be placed to gather the traffic flow here as many exploits specifically target these vulnerabilities. In Figure 7 below, a general network is represented with the sensor placement shown.

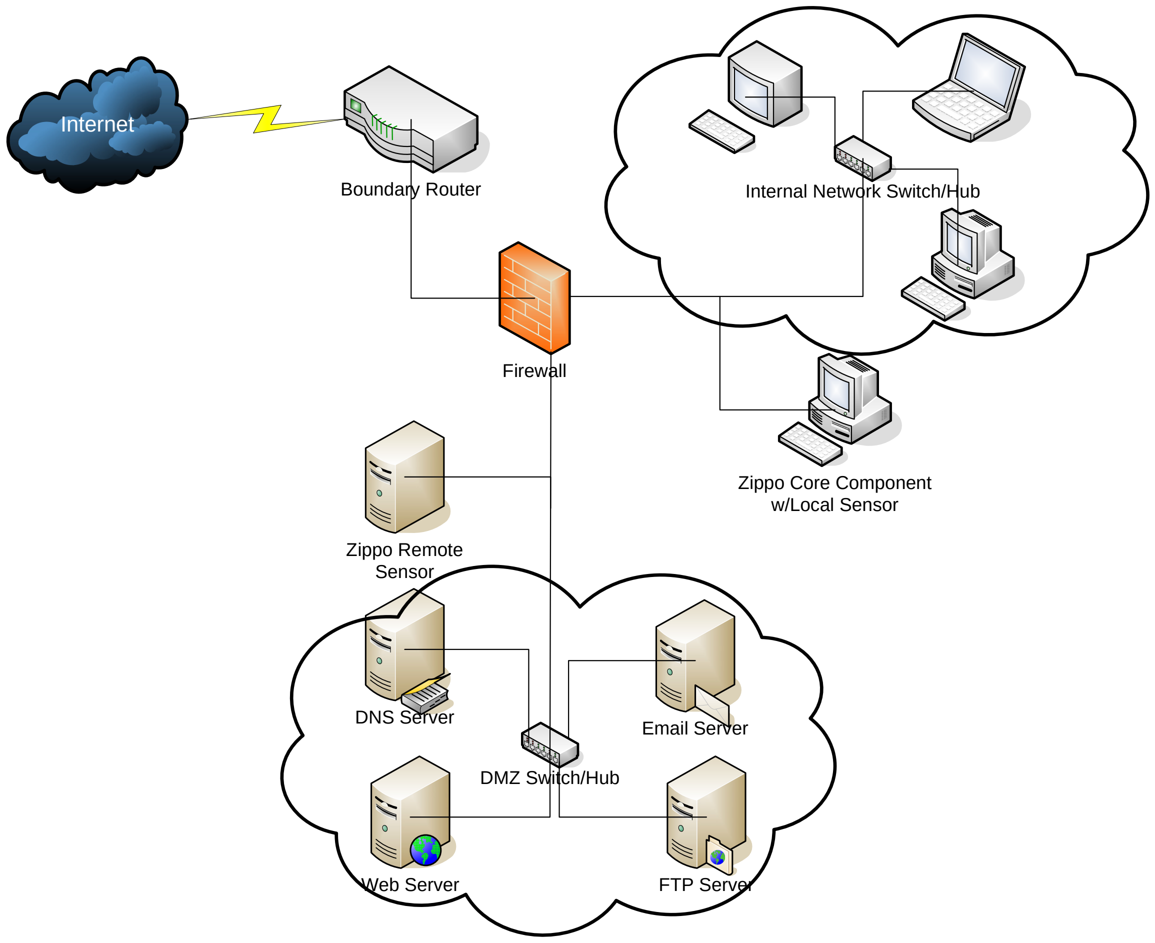

Figure 7. General network with Zippo sensors placed within the DMZ and locally on the internal network.

2. Services

A second tactic a system administrator can use to employ the Zippo sensors is to look at the services offered by his network. As shown in Figure 7 above, a DMZ resides on this network. Web servers, FTP servers and email servers, to name just a few, are all targets of opportunity for unauthorized access. If a network consists of low hanging fruit housed in a DMZ, it would behoove the system administrator to place a sensor in this section of the network. Additionally, if specific hosts inside the internal network provide easily exploitable services, these should be watched by Zippo’s sensors.

Zippo was developed to support numerous sensors placed around a network with one central core component. The thermal canyons and towers can be configured to represent a single host or a single service, based on the rules of the decision tree. This provides the system administrator with a finer granularity when monitoring the hosts on his network.

E. BUCKETS AND BALLS

The buckets and balls are the implementation of the decision tree, which determines the appearance of the thermal canyons and thermal towers, based on the decision tree’s nodes and boundary/initial conditions configured by the system administrator. Network packets are processed by the core component based on the decision tree which is modeled after expected normal traffic behavior on the network. The decision tree’s branches are defined by the topology of the network and the protocols.

The effectiveness of Zippo relies on the rules sets created for the decision tree and the boundary/initial conditions of the bucket spaces. Without adequate knowledge of normal traffic on a network, it is a true challenge to the system administrator to properly construct the decision tree to adequately screen and report on anomalous or malicious traffic.

1. The Decision Tree

In Zippo, PID instances are created by the user to define a specific decision tree to be applied to the traffic collected by the sensors. Numerous PID instances can be created and applied to the same traffic gathered from the sensors, depending on the information the system administrator is interested in (e.g. port lists, IP lists, etc.).

Zippo is configured such that the decision tree nodes can be of different types. A node can represent an IP list, a port list, new friends, matched packets or protocol lists. The leaves of the decision tree are the instances of the buckets, which are configured by the path of the decision tree branches a ball travels along to get there. Defined by the tree’s branches, the buckets represent origins, destinations, protocols and ports or any combination of these variables. Viewing the number of balls in each of the buckets at any given time, defines a state. The states change as balls move in and out of the buckets. These are the state transitions. In Figure 8 below, an example decision tree is shown to illustrate the nodes, branches and leaves (buckets).

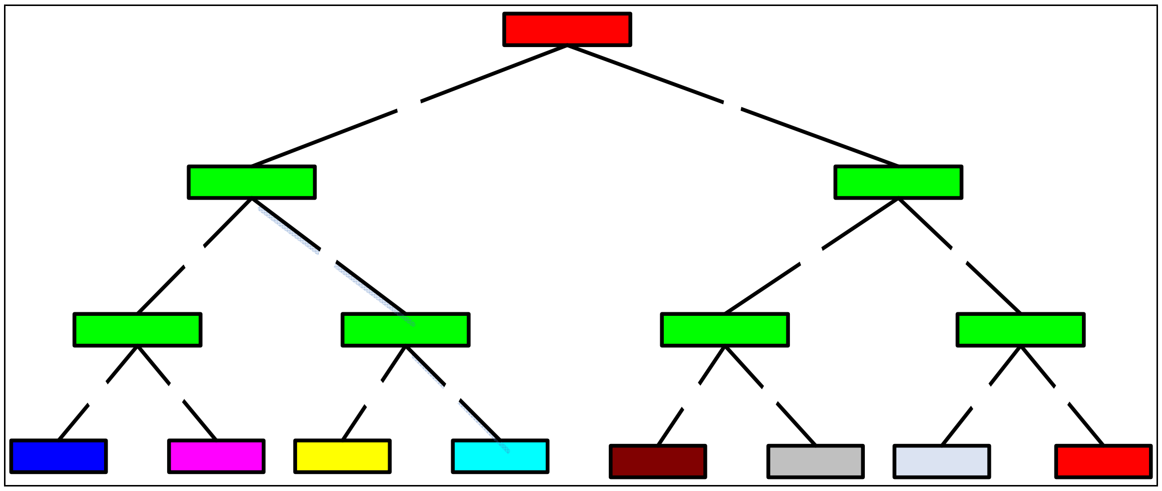

Figure 8. Decision Tree representing a PID instance. Each node represents a decision point and each bucket defines a set of rules.

In Zippo, the defining of the decision tree is analogous to creating an access control list in a firewall or the signatures in a signature-based IDS. It is the most important step the system administrator makes in configuring Zippo to his specific network.

Protocol List

It is very difficult to create a recipe for all system administrators with all types of networks to configure the decision tree. Therefore a general discussion of the decision tree with regard to configuration is presented.

Yes

The first step in configuring the decision tree is determining what "normal" traffic looks like and what level of granularity is of interest. The decision tree can then be configured based on this normal traffic. In addition the system administrator needs to determine what entities are to be monitored and to what detail. In reference to Figure 8 above, the top level (IP List node) applies the rule to the IP address of the packet. The system administrator can define this for a single IP address, a range of IP addresses or any configuration in between. If

Port List

Yes

No

Bucket 2

an IP address matches the rule in the top node, it progresses down the decision tree to be tested at each node.

The buckets at the bottom of the decision tree (or the leaves of the tree) are defined by the path the packets (or balls) take to get there. For example, bucket number 1 in figure 8 above has the identified IP address, protocol and port that are defined by the nodes in the branches above it. The system administrator can use these nodes to create a rule set just like an ACL for a firewall.

If a system administrator is checking the network for specific threats that are identified by the ports they communicate on or the protocols they use, Zippo and its decision tree provide an exceptional tool to the administrator to isolate suspicious packets. The decision tree’s nodes can be configured to look for specific malicious port use or protocol use out of the ordinary.

Along with looking for suspicious ports or protocols, a system administrator can configure the decision tree to see traffic patterns to individual servers such as a web server or a DNS server. For example, if traffic going to a web server attempts to communicate in a protocol other than http or attempts to create a connection on a port other than 80, and the administrator configures the decision tree to recognize these misguided packets, the visual display presented by the thermal canyons and thermal towers can alert the system administrator to the irregularity.

The decision tree provides a very flexible configuration tool for Intrusion Detection. It does not rely on defined signatures or abnormal behavior by users.

And with Zippo, changes to the decision tree can be made with a simple GUI and implemented immediately. The system administrator must determine what packet information is important to him and implement those choices with the decision tree's nodes. And as so eloquently presented in [MYLAV-04],

> Any model and associated system implementation depend on careful configuration of appropriate parameters to reflect the expected or assumed behavior of the underlying process to which the model is applied.

2. Bucket Conditions

Along with the definition of the decision tree nodes, the system administrator must also define the boundary and initial conditions of the buckets. The boundary conditions are the maximum and minimum number of balls allowed in each of the buckets. The larger the disparity between the boundary conditions, the larger the number of states that can be represented in any one bucket.

The initial condition of the buckets is the number of balls that start out in each bucket when the PID instance is started. Boundary and initial conditions were researched by Daniel Ettlich in [ETTL-03] and Marinovich and Walch in [MARI-04] and are presented in both references.

F. THERMAL CANYONS AND TOWERS

As discussed in previous chapters, the thermal canyons and towers are Zippo's three-dimensional representation of the network's traffic. Configuring these visual displays properly is imperative for the system administrator to adequately see the changes in the states of the network’s traffic and notice anomalies. Slidelength and windowlength are two of the defining parameters that dictate what the thermal canyons and towers look like. These are described in greater detail in Chapter IV. The ratio of the windowlength to the slidelength is known as the smoothing ratio or smoothing factor. These variables determine how the thermal canyons and thermal towers appear in the graphical display.

G. ZIPPO’S VARIABLES

Zippo’s variables are configured using the Zippo Control Center. The configuration parameters for the core component include the Number of States, Number of Time Slices, Slide Length and Smoothing Factor [ZACH-04]. These differ from the original coded variables in Therminator and are easier to enter using the interactive console.

Buckets in Zippo are defined using the interactive configuration screen for the bucket element. The defined variables include the bucket name, minimum, maximum and initial ball counts and bucket colors. Zippo automatically transfers a bucket’s color to the decision tree and to the thermal tower visualization. This gives the system administrator a clear visual picture as to what defined balls are showing in the towers.

H. SUMMARY

This chapter explored the configuration of Zippo. It presented the steps to take in making Zippo an integral part of a network’s defense in depth. It discussed how to implement Zippo in order to address the risks and vulnerabilities of the network as well as the sensor placement to protect valuable assets and address specific services. It touched on the hardware and software required to run Zippo regardless of platform.

Next, the specific variables of Zippo were discussed including how to build the decision tree and how to configure the bucket spaces. Lastly, the thermal canyons and towers were briefly presented in reference to the view they presented to the system user.

THIS PAGE INTENTIONALLY LEFT BLANK

X. MONITORING ZIPPO

A. CHAPTER OVERVIEW

This chapter presents a brief introduction to monitoring Zippo. It discusses previous experiments done by students researching Therminator and Zippo’s ability to detect malicious activity in a network. Lastly, it shows some examples of actual decision trees for different protocols as configured in Therminator.

B. RECOGNIZING ANOMALIES

As discussed numerous times in previous chapters, the visual graphic displays for Zippo are the thermal canyons and thermal towers. Each graph is important for the specific details it supplies to the system administrator.

Numerous Therminator and Zippo experiments have been carried out by Graduate students at both the Naval Postgraduate School and the University of South Carolina. The graphs and discussions presented below are the outcomes of these experiments. They are used here simply to show the reader the visual difference between normal network traffic and anomalous network traffic. Additionally, graphs will be shown that depict the differences in slidelength and windowlength as discussed previously.

In Figure 9 below, a thermal canyon is presented from reference [MCEATHERM-04]. It shows normal traffic in a network. Figure 10, just below it shows the original traffic on the same network, but it introduces one anomalous packet to the mix. The aggravation to the visual display is very obvious.

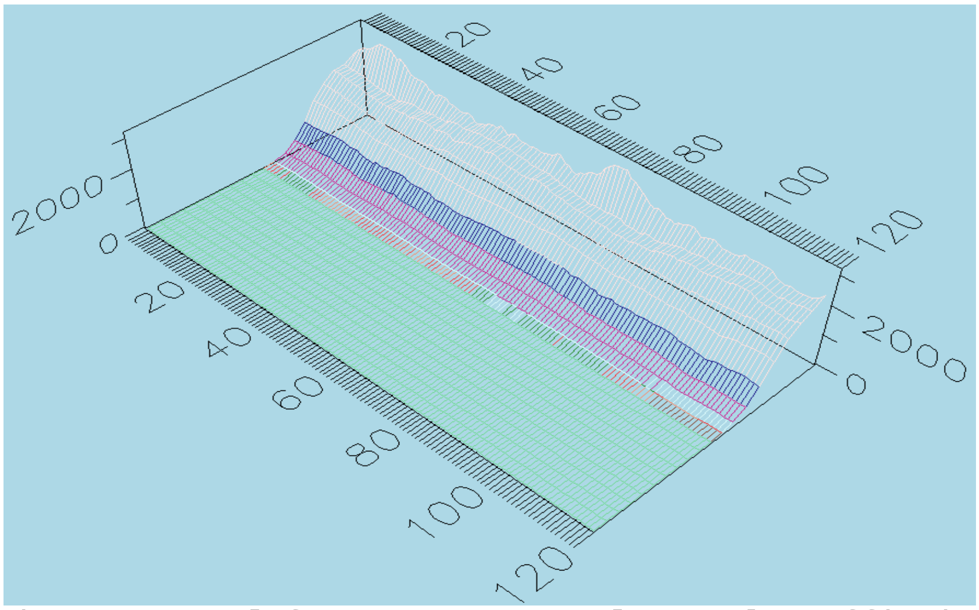

Figure 9. [After MCEATHERM-04] Normal traffic in a network. This is a Therminator display of 200,000 packets sent to known web servers.

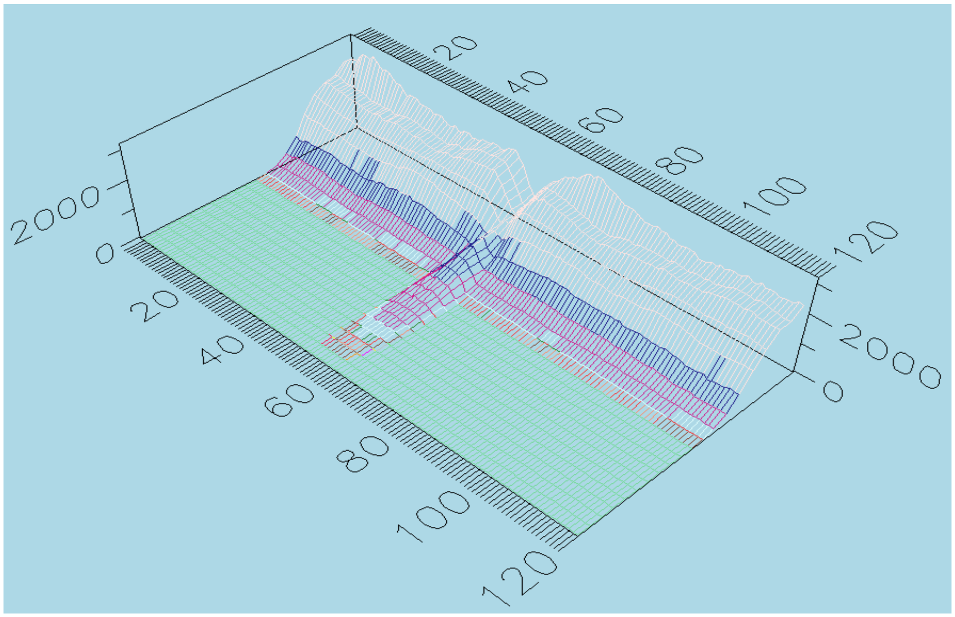

Figure 10. [After MCEATHERM-04] Same network, same 200,000 packets with the introduction of one anomalous packet.

Comparing the graphs in Figures 9 and 10, it becomes quite obvious that a traffic anomaly has taken place in this network. The variables in Zippo must be configured correctly for malicious traffic to be so flagrant.

Figure 11 below shows an example of an ICMP anomaly. In this instance, the spike in the graph shows an increased amount of ICMP echo requests.

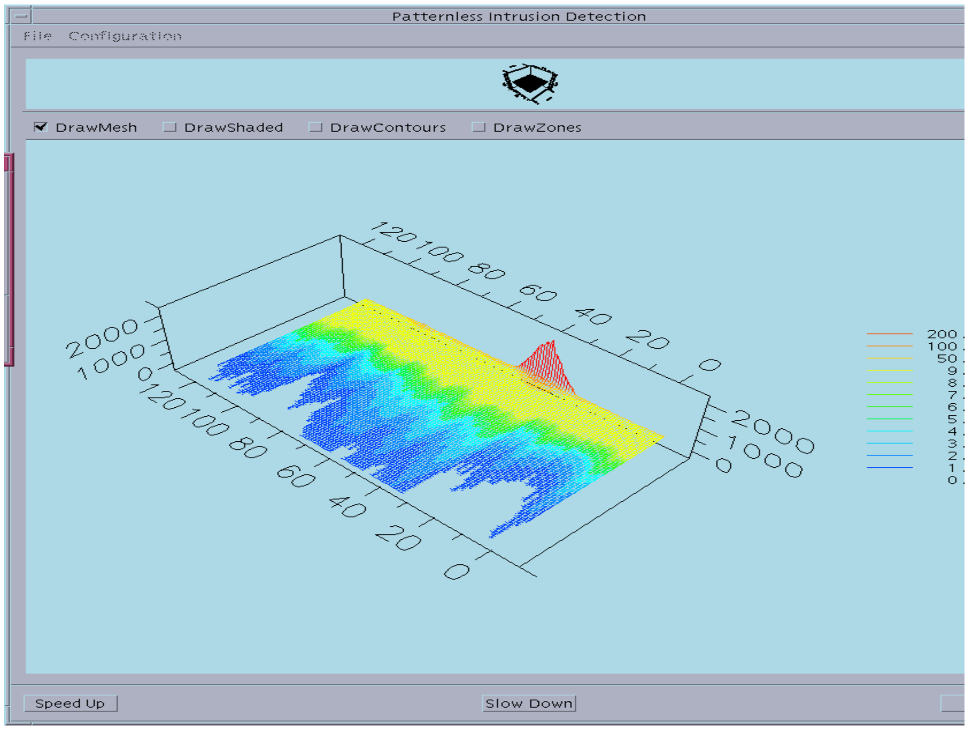

Figure 11. [After DONA-01] Graphic display showing an abnormal amount of ICMP traffic.

Figures 12-14 show different attacks applied to a network with Zippo installed. These graphs were taken from reference [KAHWAI–04] and represent work she did experimenting with using Zippo with distributed sensors. Each of the graphs display a different attack with a different PID instance defined for each. The PID instance was reflective of the type of packets that could be expected for the particular attacks.

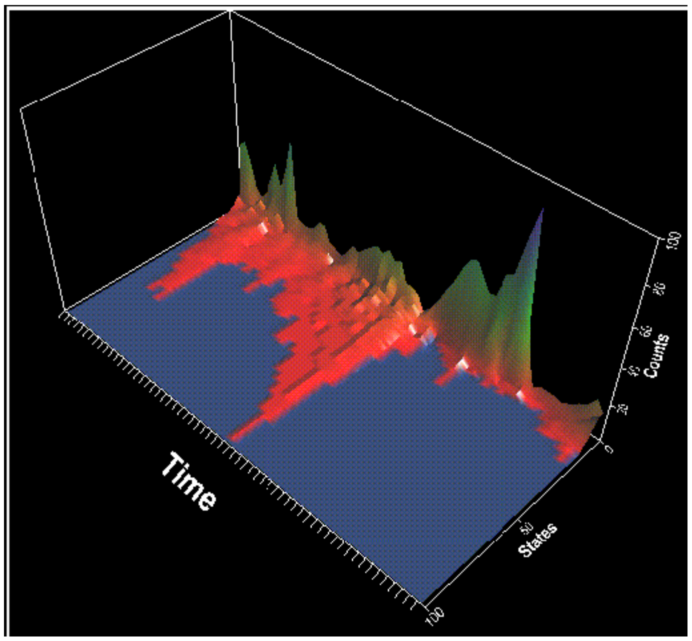

Figure 12. [After KAHWAI-04] Graphic display of the Thermal Canyon in Zippo during a Mailbomb attack. The PID instance reflected in this graphic is for SMTP traffic.

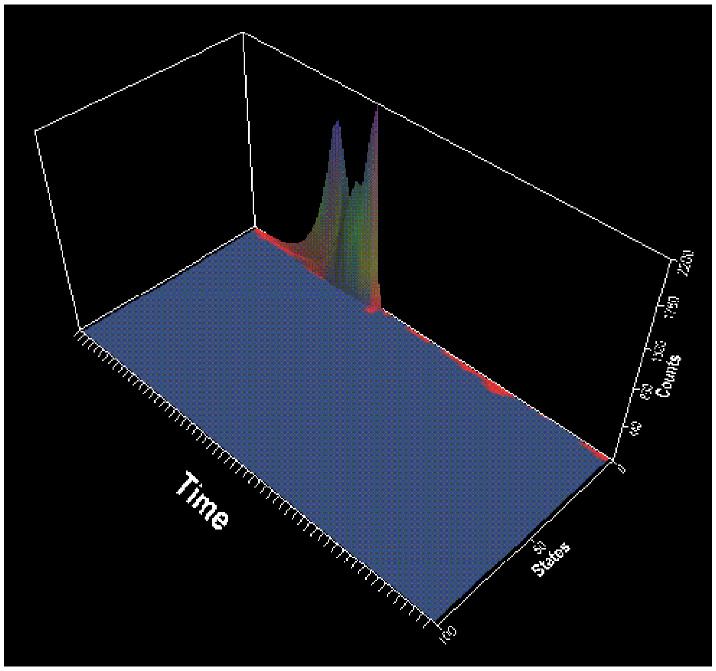

Figure 13. [After KAHWAI-04] Graphic Display of the Thermal Canyon in Zippo during a Smurf Attack. The PID instance reflected in this graphic is for ICMP traffic.

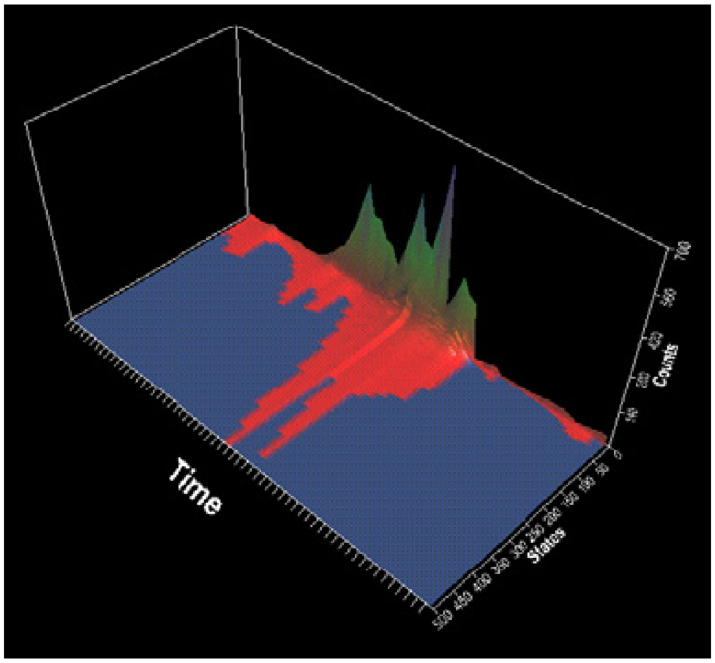

Figure 14. [After KAHWAI-04] Graphic Display of the Thermal Canyon in Zippo during an Apache2 attack. Since this would be used against a web server, the PID instance reflected in this graphic is for http traffic.

C. DECISION TREES

The following figures and tables show decision trees developed by Cheng Kah Wai in [KAHWAI-04] to represent the PID instances of the thermal canyon graphic displays presented in Figures 12 – 14 above. The decision trees are developed for specific protocols. These protocols were chosen as they are the protocols used in the attack that were used to test the Zippo application. These are presented as a representation of Zippo’s decision trees and how they can be configured for three different protocols.

The tables that follow each figure explain in greater detail what the rule sets are at each node of the decision tree. This clarifies for the reader how a ball travels through the branches of a tree and ultimately into the buckets.

There are two important aspects the reader should notice in reviewing the decision trees:

- The configuration of the decision trees is not unique beyond what is required to configure the protocol. In other words, the decision tree looks at what port number is used by the application in question and what IP address is assigned to the server. This information can be directly taken from the security plan
- The exchange of balls for normal traffic will occur primarily on one side of the decision tree. This leaves the other half of the tree available to capture balls associated with anomalous traffic.

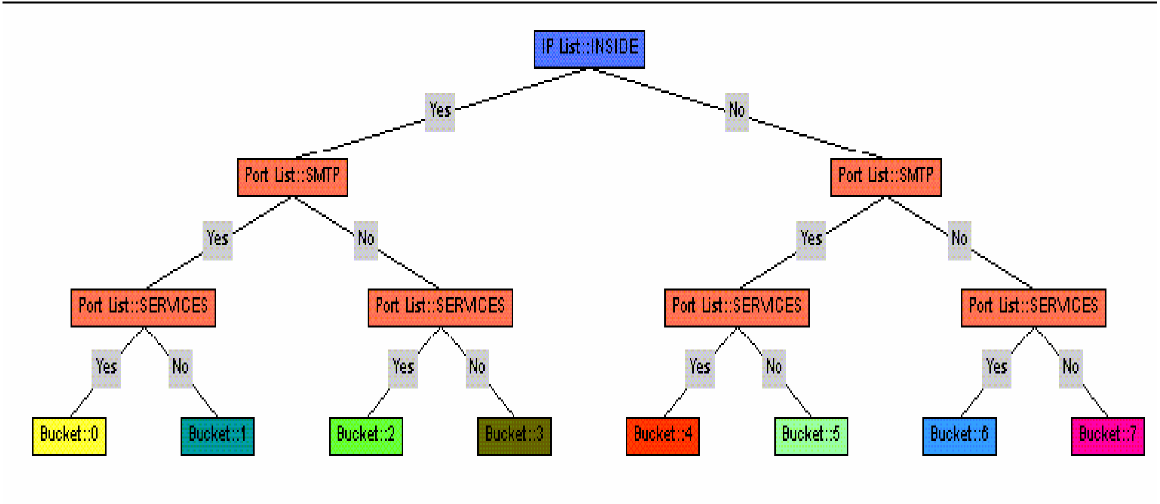

Figure 15. [After KAHWAI-04] A decision tree for the SMTP PID instance. This is the tree used in the mailbomb attack shown in Figure 12.

Bucket No.	Classification
0	N.A.
1	Insider IP address with TCP ports no. 25, 110, 113 or 161
2	Insider IP address with TCP port no. lower than 1024, excluding 25, 110, 113 and 161.
3	Insider IP address that does not have TCP port no. lower than 1024.
4	N.A.
5	Outsider IP address with TCP ports no. 25, 110, 113 or 161.
6	Outsider IP address with TCP port no. lower than 1024, excluding 25, 110, 113 and 161.
7	Outsider IP address that does not have TCP port no. lower than 1024.

Table 3. [After KAHWAI-04] Description of bucket definitions for SMTP PID instance in Figure 15.

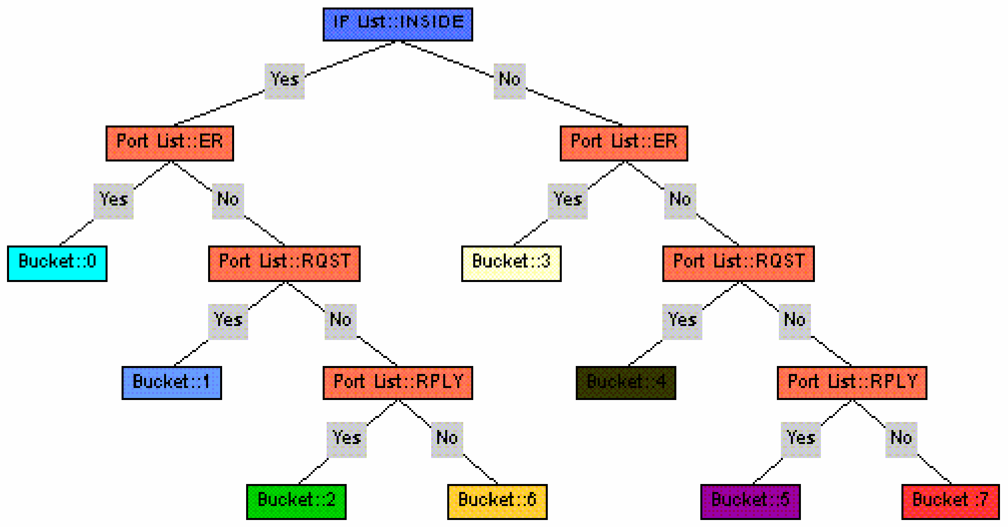

Figure 16. [After KAHWAI-04] A decision tree for the ICMP PID instance. This is the tree used in the Smurf attack shown in Figure 13.

Bucket No.	Classification
0	Insider IP address with ICMP type 3, 4, 5, 11 or 12.
1	Insider IP address with ICMP type 8 or 17.
2	Insider IP address with ICMP type 0 or 18.
3	Outsider IP address with ICMP type 3, 4, 5, 11 or 12.
4	Outsider IP address with ICMP type 8 or 17.
5	Outsider IP address with ICMP type 0 or 18.
6	Insider IP address that does not have ICMP type 0, 3, 4, 5, 8, 11, 12, 17 or 18.
7	Outsider IP address that does not have ICMP type 0, 3, 4, 5, 8, 11, 12, 17 or 18.

Table 4. [After KAHWAI-04] Description of bucket definitions for ICMP PID instance in Figure 16.

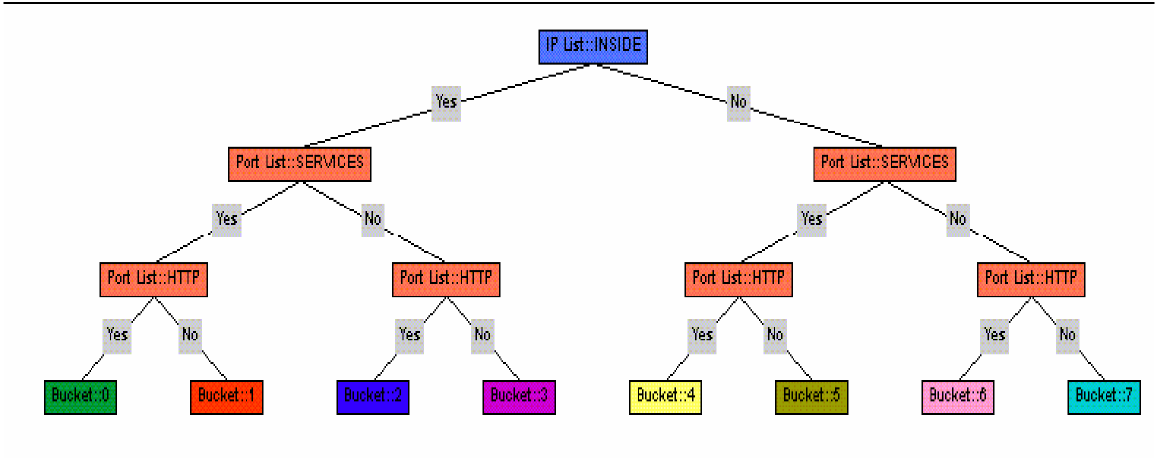

Figure 17. [After KAHWAI-04] A decision tree for the HTTP PID instance. This is the tree used in the Apache 2 attack shown in Figure 14.

Bucket No.	Classification
0	N.A
1	Insider IP address with TCP port no. lower than 1024, excluding 80 and 443.
2	Insider IP address with TCP port no. 80 or 443.
3	Insider IP address that does not TCP port no. lower than 1024.
4	N.A
5	Outsider IP address with TCP port no. lower than 1024, excluding 80 and 443.
6	Outsider IP address with TCP port no. 80 or 443.
7	Outsider IP address that does not TCP port no. lower than 1024.

Table 5. [After KAHWAI-04] Description of bucket definitions for HTTP PID instance in Figure 17.

D. SUMMARY

The graphs, decision trees and tables presented in this chapter, were used to introduce the reader to the graphical three-dimensional visualization of Zippo’s monitoring engine. The graphs were created by previous Master’s students at the Naval Postgraduate School when researching the ability of Therminator and Zippo to detect anomalous network traffic. The decision trees and tables presented at the end of the chapter are good visual representations of what the rule sets logically look like.

THIS PAGE INTENTIONALLY LEFT BLANK

XI. SUMMARY AND FUTURE RESEARCH

A. SUMMARY

This study’s purpose was to determine a methodology for implementing a Patternless Intrusion Detection System known as Zippo. Research and interviews were conducted at a Federal Government organization and many of their ideas and network concerns were presented in this study.

To effectively implement Zippo, it is important for the system administrator to understand how it works. Chapter II was a brief review of the TCP and IP protocols which are necessary to understand for proper configuration of the decision trees in Zippo. Although system administrators are probably well aware of how these protocols work, this chapter was written to explain the basics to any reader of the thesis.

Along with understanding the building blocks of network communication, it is also important to have a general overview of the evolution of Zippo. This started in Chapter III with an introduction to Intrusion Detection Systems including the different configurations and different types of rule-bases, to include signature-based, anomaly-based and the latest, patternless based on statistical analysis and thermodynamics principles. This is the basis for Therminator.

Therminator, the predecessor of Zippo, was presented in Chapter IV to explain how the Patternless Intrusion Detection System works. It goes into great detail of the key components of the PIDS, including the core component, the sensors, the bucket and ball decision trees and the visual graphic representation of the network traffic.

Without a basic understanding of Therminator and patternless intrusion detection, it would be difficult for the reader to understand how Zippo evolved.

Zippo, the Intrusion Detection System which is the focus of this study was presented in Chapter V. Here, the differences between Therminator and Zippo were compared and the improvements and changes that Zippo implemented were presented. To conclude the chapter, the architecture of Zippo and the process of implementing it in a network were discussed.

Chapters VI – VIII departed from the Zippo topic and focused on the assets that should be in place in an organization that wants to implement an Intrusion Detection System. These chapters focus on administrative as well as operational aspects of a network. The necessary documents suggested in these chapters were based on experience and research with other organizations. In order to protect a network, a system administrator must know what a network is made of and what it is doing. These aspects of a network can be found in a thoroughly developed network topology and inventory and a well-written and adhered to security plan. Having detailed documents in place at an organization and having the policies and procedures adhered to are invaluable to a system administrator trying to secure a network.

Chapters IX and X presented a more detailed look at Zippo. Chapter IX went into great detail of how to develop a decision tree and how to determine the rule sets for the nodes. In this study, a set list of standards to create a decision tree could not be developed as was originally desired. There are too many factors in any one network that determine the optimal configuration of the decision tree. Trying to create a template for any network has proven to be too weighty. Therefore, information is provided in Chapter IX to assist the system administrator in developing the core components, graphical displays and sensors for a general network configuration.

Chapter X provided examples, from previous Graduate students' thesis, of thermal canyons and decision trees created to detect specific attacks. The decision trees in the second half of the chapter are a good reference when configuring a decision tree for the specific protocols mentioned. This chapter was included to assist the reader in visualizing what Zippo is capable of.

Overall, the objective of the thesis research was met. However, the original desire to create a template for any given network for the decision trees in Zippo, proved too difficult considering all the factors that influence the configuration and protection of a network.

B. FUTURE WORK

Future work for Zippo should include research into developing, protocol by protocol, instance by instance, optimal decision trees for different purposes. Perhaps optimal decision trees can be created for well-known attacks.

Additionally, Zippo can be researched further on robust networks. In the short amount of time the author has worked with the program it seems to provide a very capable security application to assist a system administrator in successfully securing his network.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[BEJT-05] Bejtlich, R., *The Tao of Network Security Monitoring, Beyond Intrusion Detection,* Addison Wesley, New York, 2005

[CROTH-03] Crothers, T., *Implementing Intrusion Detection Systems, A Hands-On Guide for Securing the Network,* Wiley Publishing Inc., Indiana, 2003

[DONA-01] Donald, S. and McMillen, R., “Therminator 2: Developing a Real Time Thermodynamic Based Patternless Intrusion Detection System”, Master’s Thesis, Naval Postgraduate School, Monterey, California, 2001

[ETTL-03] Ettlich, D., “Therminator: Configuring the Underlying Statistical Mechanics Model”, Master’s Thesis, Naval Postgraduate School, Monterey, California, 2003

[KAHWAI-04] Kah Wai, C., “Distributed Deployment of Therminators in the Network”, Master’s Thesis, Naval Postgraduate School, Monterey, California, 2004

[MARI-04] Marinovich, J. and Walch, S., “Analysis of Initial and Boundary Conditions in Therminator Conversation Exchange Dynamics”, Master’s Thesis, Naval Postgraduate School, Monterey, California, 2004

[MCEAC-04] McEachen, J., Zachary, J. and Ford, D., “Therminator, A Transformational Enabler for FORCEnet”, *CHIPS,* 2004

[MCEATHERM-04] McEachen, J., Therminator Presentation, Naval Postgraduate School, 2004

[MYLAV-04] Mylavarapu, S., Walch, S., Marinovich, J., Zachary, J., McEachen, J. and Ford, D., “Monitoring Conversation Exchange Dynamics for Detection of Epidemic-Style Computer Network Attacks”, 2004

[NIST-98]	NIST Special Publication 800-18, Guide for Developing Security Plans for Information Technology Systems, 1998
[NORETAL-03]	Northcutt, S., Zeltser, L., Winters, S., Frederick, K. and Ritchey, R., *Inside Network Perimeter Security, The Definitive Guide to Firewalls, VPNs, Routers and Intrusion Detection Systems* SANS GIAC, New Riders Publishing, Indianapolis, 2003
[NORTH-03]	Northcutt, S. and Novak, J., *Network Intrusion Detection,* SANS GIAC, New Riders Publishing, Indianapolis, 2003
[SECCOG-04]	Secure Cognition Incorporated www.securecognition.com 15 September 2004
[ZACH-04]	Zachary, J., "Zippo: A Robust and Portable Network Anomaly Detection System", 2004

BIBLIOGRAPHY

Ahlm, E., “Vigilar: Is Intrusion Prevention Changing Information Security?”, 2004

Debar, H., Dacier, M. and Wespi, A., “Towards a Taxonomy of Intrusion-Detection System”, *Computer Networks 31*, 1999

Lasser, J., “Intrusion Detection: Implementing Snort in a Production Environment”, *;login: The Magazine of Usenix & Sage*, 2001

McEachen, J., Zachary, J. and Ettlich, D., “Differentiating Network Conversation Flow for Intrusion Detection and Diagnostics

Yee, A., “The Intelligent IDS: Next Generation Network Intrusion Management Revealed”, 2003

Zachary, J., McEachen, J. and Ettlich, D., “Conversation Exchange Dynamics for Real-Time Network Monitoring and Anomaly Detection”, 2004

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Naval School, Civil Engineers Corps Officers Code C35
   Naval Construction Battalion Center
   Port Hueneme, California

4. Dr. John McEachen
   Naval Postgraduate School
   Monterey, California

5. Dr. Alex Bordetsky
   Naval Postgraduate School
   Monterey, California

6. Dr. Dan Boger
   Naval Postgraduate School
   Monterey, California